1/5/1 (Item 1 from file: 351) Links
Fulltext available through: Order File History
Derwent WPI

0012791078 & & *Drawing available*
WPI Acc no: 2002-646776/**200270**
XRPX Acc No: N2002-511517
Scheduling correctness checking method for circuit, involves executing symbolic simulation for extracting loop invariant term for determining sufficient set of non-cyclic thread
Patent Assignee: NEC CORP (NIDE)
Inventor: ASHAR P; BHATTACHARYA S; GUPTA A; RAGHUNATHAN A; SUBURAJITTO B

Patent Family (3 patents, 2 & countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	Type
JP 2001142937	A	20010525	JP 2000106543	A	20000407	200270	B
US 6745160	B1	20040601	US 1999414815	A	19991008	200436	E
US 20040148150	A1	20040729	US 1999414815	A	19991008	200450	E
			US 2004756303	A	20040114		

Priority Applications (no., kind, date): US 1999414815 A 19991008; US 2004756303 A 20040114

Patent Details

Patent Number	Kind	Lan	Pgs	Draw	Filing Notes	
JP 2001142937	A	JA	37	1		
US 20040148150	A1	EN			Continuation of application	US 1999414815
					Continuation of patent	US 6745160

Alerting Abstract JP A
NOVELTY - A symbolic simulation is executed for extracting the loop invariant term for determining the sufficient set of a non-cyclic thread, when a loop is inside the circuit, and the equivalency of non-cyclic thread is proved.
USE - For checking correctness of scheduling of circuit.
ADVANTAGE - The scheduling correctness of circuit is checked corresponding to behaviors description of circuit, using simple technique.
DESCRIPTION OF DRAWINGS - The figure shows a block diagram of scheduling correctness checking system.
(Drawing includes non-English language text).

Title Terms /Index Terms/Additional Words: SCHEDULE; CORRECT; CHECK; METHOD; CIRCUIT; EXECUTE; SYMBOL; SIMULATE; EXTRACT; LOOP; INVARIANT; TERM; DETERMINE; SUFFICIENT; SET; NON; CYCLIC; THREAD

Class Codes

International Patent Classification

IPC	Class Level	Scope	Position	Status	Version Date
G06F-017/50			Main		"Version 7"
G06F-0017/50	A	I		R	20060101
G06F-0017/50	C	I		R	20060101

US Classification, Issued: 70314, 70314, 70313, 70315, 70316, 7161, 7168

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－142937
（P2001－142937A）
(43)公開日　平成13年５月25日(2001.5.25)

(51)Int.Cl.⁷	識別記号	Ｆ Ｉ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　17/50	６６４	Ｇ０６Ｆ　17/50	６６４Ｇ　５Ｂ０４６

審査請求　未請求　請求項の数43　ＯＬ　（全 37 頁）

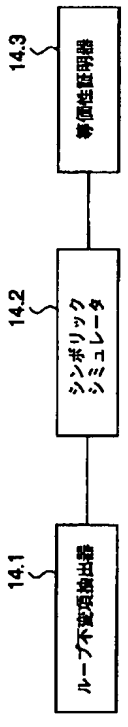
(21)出願番号　特願2000－106543(P2000－106543)
(22)出願日　平成12年４月７日(2000.4.7)
(31)優先権主張番号　０９／４１４８１５
(32)優先日　平成11年10月８日(1999.10.8)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　プラナブ・アシャー
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内
(74)代理人　100097157
弁理士　桂木　雄二

最終頁に続く

(54)【発明の名称】　回路のスケジューリング正当性チェック方法及びスケジュール検証方法

(57)【要約】
【課題】　回路のスケジューリングの正当性をチェックする方法、及び、回路のビヘイビア記述に対して回路のスケジュールを検証する方法を実現する。
【解決手段】　回路に対するスケジュールはビヘイビア記述から得られる。回路のスケジューリングの正当性をチェックする方法は、ループが回路内にあるときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出し、ループ不変項を抽出するためにシンボリックシミュレーションを実行し、非巡回スレッドの等価性を証明する。回路のビヘイビア記述に対して回路のスケジュールを検証する方法は、スケジュールからループを含む可能性のある実行のスケジュールスレッドを選択し、ビヘイビア記述から対応するビヘイビアスレッドを識別し、スケジュールスレッド及びビヘイビアスレッドの無条件等価性を証明し、実行のすべてのスレッドについて以上を繰り返す。



1

【特許請求の範囲】

【請求項１】　回路に対するスケジュールがビヘイビア記述から得られる場合の当該回路のスケジューリングの正当性をチェックする方法において、

（ａ）ループが回路内にあるときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出するステップと、

（ｂ）前記ループ不変項を抽出するためにシンボリックシミュレーションを実行するステップと、

（ｃ）前記非巡回スレッドの等価性を証明するステップと、

からなることを特徴とする回路スケジューリング正当性チェック方法。

【請求項２】　前記ビヘイビア記述は、サイクル境界の導入によって変換されることを特徴とする請求項１記載の方法。

【請求項３】　前記ビヘイビア記述は、演算並べ替えによって変換されることを特徴とする請求項１記載の方法。

【請求項４】　前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換されることを特徴とする請求項１記載の方法。

【請求項５】　前記ビヘイビア記述は、演算の投機実行によって変換されることを特徴とする請求項１記載の方法。

【請求項６】　前記ステップ（ｃ）は、シンボリックシミュレーションを用いて実行されることを特徴とする請求項１記載の方法。

【請求項７】　回路のビヘイビア記述に対して回路のスケジュールを検証する方法において、

（ａ）前記スケジュールから、ループを含む可能性のある実行のスケジュールスレッドを選択するステップと、

（ｂ）前記ビヘイビア記述から、対応するビヘイビアスレッドを識別するステップと、

（ｃ）スケジュールスレッド及びビヘイビアスレッドの無条件等価性を証明するステップと、

（ｄ）実行のすべてのスレッドについて前記ステップ（ａ）～（ｃ）を繰り返すステップと、

からなることを特徴とする回路スケジュール検証方法。

【請求項８】　前記スケジュールは、スケジュール状態遷移グラフとして指定されることを特徴とする請求項７記載の方法。

【請求項９】　前記ビヘイビアは、ビヘイビア状態遷移グラフとして指定されることを特徴とする請求項７記載の方法。

【請求項１０】　前記ステップ（ｃ）は、

（ｉ）前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、

（ｉｉ）前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、

からなることを特徴とする請求項７記載の方法。

2

【請求項１１】　回路のビヘイビア記述に対して回路のスケジュールを検証する方法において、

（ａ）スケジュールをスケジュール状態遷移グラフとして指定するステップと、

（ｂ）回路のビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、

（ｃ）前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するステップと、

（ｄ）前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを識別するステップと、

（ｅ）前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、

（ｆ）前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、

（ｇ）実行のすべてのスレッドについて前記ステップ（ｃ）～（ｆ）を繰り返すステップと、

からなることを特徴とする回路スケジュール検証方法。

【請求項１２】　前記ステップ（ｆ）は、

（ｉ）前記ビヘイビア状態遷移グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記ビヘイビア構造グラフ内のすべてのノードを含む順序セットａｒｒ１を作成するステップと、

（ｉｉ）前記ビヘイビア構造グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記スケジュール構造グラフ内のすべてのノードを含む順序セットａｒｒ２を作成するステップと、

（ｉｉｉ）ａｒｒ１をたどり、ビヘイビア構造グラフ内の基底変数を識別するステップと、

（ｉｖ）ビヘイビア構造グラフ内の非基底変数を基底変数で表すステップと、

（ｖ）スケジュール構造グラフ内の入力ノードに対する等価性リストを構成するステップと、

（ｖｉ）ａｒｒ２をたどり、ａｒｒ２内の各ノードを処理して、スケジュール構造グラフの入力からスケジュール構造グラフの出力へ等価性リストを伝搬させるステップと、

（ｖｉｉ）ｕをビヘイビア構造グラフ内の信号の識別子とし、ｃを等価性の条件を表す二分決定ダイヤグラムであるとして、各等価性リスト内のエントリは対（ｕ，ｃ）であり、ビヘイビア構造グラフ内の対応する出力ノードで等価性が確定したかどうか、及び、対応する条件ｃがａｒｒ２内のプライマリ出力ノードに対するトートロジーであるかどうかをチェックするステップと、

（ｖｉｉｉ）ａｒｒ２内のすべての出力ノードについて前記ステップ（ｖｉｉ）を繰り返すステップと、

（ｉｘ）すべての出力ノードが等価であることがわかった場合に等価性を見つけるステップと、
からなることを特徴とする請求項１１記載の方法。
【請求項１３】　実行の巡回スレッドを有する可能性のある回路のスケジュールと該回路のビヘイビアとの間の等価性を検証する方法において、
（ａ）スケジュールをスケジュール状態遷移グラフとして表現するステップと、
（ｂ）ビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、
（ｃ）前記スケジュール状態遷移グラフ内の強連結成分を識別するステップと、
（ｄ）各強連結成分内の終了ノードを識別するステップと、
（ｅ）前記スケジュール状態遷移グラフをつぶして、前記強連結成分を通らないサブパスを併合するステップと、
（ｆ）以前に選択されていないパスを選択するステップと、
（ｇ）前記ステップ（ｆ）で選択されたパスに対する構造ＲＴＬ回路を取得するステップと、
（ｈ）選択されたパスを列挙するのに必要なすべての状態遷移決定をカプセル化するパスシグナルを生成するための回路を構造ＲＴＬ回路に追加するステップと、
（ｉ）パスシグナルを用いて、制約されたシンボリックシミュレーションを実行してビヘイビア状態遷移グラフ内の対応するパスを識別し、該パスに対する構造ＲＴＬ回路を取得するステップと、
（ｊ）選択されたパスにおいて、以前に選択されていない強連結成分を選択するステップと、
（ｋ）選択されたパス内の選択された強連結成分に対する不変項を、対応セットのリストとして抽出するステップと、
（ｌ）対応セットのリストから１つの対応セットを選択するステップと、
（ｍ）選択された対応セットが、前のシンボリックシミュレーションの強連結成分カットにおいて得られる変数対応より小さい場合に、シンボリックシミュレーションを再実行するステップと、
（ｎ）対応セットのリスト内の各対応セットについて前記ステップ（ｉ）～（ｍ）を繰り返すステップと、
（ｏ）出力等価性条件が、パス条件以外の条件付きであるかどうかをテストするステップと、
（ｐ）前記ステップ（ｏ）で前記出力等価性が条件付きである場合に非等価性を報告してこの方法を終了するステップと、
（ｑ）選択されたパス内のすべての強連結成分について前記ステップ（ｊ）～（ｐ）を繰り返すステップと、
（ｒ）終了点が高々３度現れるようにルートからシンクへのすべてのパスについて前記ステップ（ｆ）～（ｑ）

を繰り返すステップと、
からなることを特徴とする前記回路のスケジュールとビヘイビアとの間の等価性を検証する方法。
【請求項１４】　前記ステップ（ｉ）の制約されないシンボリックシミュレーションは、
（ｉ）ビヘイビア状態遷移グラフの始状態を許容パスリストに割り当てるステップと、
（ｉｉ）許容パスリスト内で以前に訪れていない状態を選択するステップと、
（ｉｉｉ）ビヘイビア構造ＲＴＬを生成するステップと、
（ｉｖ）非解釈シンボリックシミュレーションを実行して、スケジュール構造ＲＴＬ及びビヘイビア構造ＲＴＬ内の対応する信号を識別するステップと、
（ｖ）遷移条件とパスシグナルの論理積がゼロでない場合に、状態S_jの新しいコピーを許容パスに追加するステップと、
（ｖｉ）S_iからS_jへの各出遷移ごとに前記ステップ（ｖ）を繰り返すステップと、
（ｖｉｉ）許容パス内に残る訪れていない状態のみが終状態のインスタンスとなるまで、すべての訪れていない状態について前記ステップ（ｉｉｉ）～（ｖｉ）を繰り返すステップと、
からなるプロセスを用いて実行されることを特徴とする請求項１３記載の方法。
【請求項１５】　前記ステップ（ｋ）において、不変項は、各ループごとに、
（ｉ）各カットが前記ループの各実行の境界における変数値を表すような、スケジュール内のパスの構造ＲＴＬ回路内の３個のカットを識別するステップと、
（ｉｉ）ビヘイビアにおけるパスの構造ＲＴＬ回路内の対応するカットを識別して、第１と第２のカットの間のサブ回路と、第２と第３のカットの間のサブ回路が同型であることをチェックするステップと、
（ｉｉｉ）スケジュール及びビヘイビアのＲＴＬ回路における対応するカットの各対における変数どうしの間の等価関係を識別するステップと、
（ｉｖ）最後のカットと最後の前のカットとの間の等価関係が同一であるかどうかをチェックするステップと、
（ｖ）前記ステップ（ｉｖ）の関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットである場合、最後の前のカットにおける等価関係を破棄し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、前記ステップ（ｉｉｉ）から繰り返すステップと、
（ｖｉ）前記ステップ（ｉｖ）の関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットでない場合、最後の前のカットにおける等価関係を、等価関係セットの集合に追加し、１つ以上のループ実行について２つのＲＴＬ回

路を展開して、前記ステップ（ｉｉｉ）から繰り返すステップと、

（ｖｉｉ）前記ステップ（ｉｖ）の関係が同一である場合、最後のカットにおける等価関係を、等価関係セットの集合に追加するステップと、

（ｖｉｉｉ）等価関係セットの集合内で、他のエントリのスーパーセットであるすべてのエントリを削除するステップと、

（ｉｘ）等価関係セットの最終集合を、不変項の所望の集合として指定するステップと、

からなるプロセスを用いてループから抽出されることを特徴とする請求項１３記載の方法。

【請求項１６】　回路に対するスケジュールがビヘイビア記述から得られ、回路のスケジューリングの正当性をチェックするシステムにおいて、

ループが存在するときに非巡回スレッドの十分なセットを決定するループ不変項抽出器と、

前記ループ不変項を抽出するシンボリックシミュレータと、

非巡回スレッドの等価性を証明する等価性証明器と、

からなることを特徴とする回路のスケジューリングの正当性をチェックするシステム。

【請求項１７】　前記ビヘイビア記述は、サイクル境界の導入によって変換されることを特徴とする請求項１６記載のシステム。

【請求項１８】　前記ビヘイビア記述は、演算並べ替えによって変換されることを特徴とする請求項１６記載のシステム。

【請求項１９】　前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換されることを特徴とする請求項１６記載のシステム。

【請求項２０】　前記ビヘイビア記述は、演算の投機実行によって変換されることを特徴とする請求項１６記載のシステム。

【請求項２１】　回路のビヘイビア記述に対して回路のスケジュールを検証するシステムにおいて、

スケジュールをスケジュール状態遷移グラフとして指定するスケジュール状態遷移グラフジェネレータと、

回路のビヘイビアをビヘイビア状態遷移グラフとして指定するビヘイビア状態遷移グラフジェネレータと、

前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するスケジュールスレッドセレクタと、

前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを選択するビヘイビアスレッドセレクタと、

前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するコンバータと、

前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックする等価性チェッカと、

からなることを特徴とする回路のビヘイビア記述に対して回路のスケジュールを検証するシステム。

【請求項２２】　回路のスケジューリングの正当性をチェックするための、

プロセッサ及びメモリを有するコンピュータシステムにおいて、

回路に対するスケジュールは、ビヘイビア記述から得られ、

前記メモリは、前記コンピュータシステムが前記チェックを実行することを可能にする命令を含み、該命令は、

ループが存在するときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出する命令と、

ループ不変項を抽出するためのシンボリックシミュレーションの命令と、

非巡回スレッドの等価性を証明する命令と、

を含むことを特徴とする回路のスケジューリングの正当性をチェックするためのコンピュータシステム。

【請求項２３】　前記ビヘイビア記述は、サイクル境界の導入によって変換されることを特徴とする請求項２２記載のコンピュータシステム。

【請求項２４】　前記ビヘイビア記述は、演算並べ替えによって変換されることを特徴とする請求項２２記載のコンピュータシステム。

【請求項２５】　前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換されることを特徴とする請求項２２記載のコンピュータシステム。

【請求項２６】　前記ビヘイビア記述は、演算の投機実行によって変換されることを特徴とする請求項２２記載のコンピュータシステム。

【請求項２７】　回路のビヘイビア記述に対して回路のスケジュールを検証するための、プロセッサ及びメモリを有するコンピュータシステムにおいて、

前記メモリは、前記コンピュータシステムが前記検証を実行することを可能にする命令を含み、該命令は、

スケジュールをスケジュール状態遷移グラフとして指定する命令と、

回路のビヘイビアをビヘイビア状態遷移グラフとして表現する命令と、

前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択する命令と、

前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを選択する命令と、

前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換する命令と、

前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックする命令と、

実行のすべてのスレッドについて繰り返す命令と、

を含むことを特徴とする、回路のビヘイビア記述に対し

て回路のスケジュールを検証するためのコンピュータシステム。

【請求項２８】　回路のビヘイビア記述に対して回路のスケジュールを検証するための、プロセッサ及びメモリを有するコンピュータシステムにおいて、

前記メモリは、前記コンピュータシステムが、

（ａ）スケジュールをスケジュール状態遷移グラフとして指定するステップと、

（ｂ）回路のビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、

（ｃ）前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するステップと、

（ｄ）前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを識別するステップと、

（ｅ）前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、

（ｆ）前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、

（ｇ）実行のすべてのスレッドについて前記ステップ（ｃ）～（ｆ）を繰り返すステップと

を実行することを可能にする命令を含むことを特徴とする、回路のビヘイビア記述に対して回路のスケジュールを検証するためのコンピュータシステム。

【請求項２９】　前記命令は、前記コンピュータシステムが、

（ｉ）前記ビヘイビア状態遷移グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記ビヘイビア構造グラフ内のすべてのノードを含む順序セットａｒｒ１を作成するステップと、

（ｉｉ）前記ビヘイビア構造グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記スケジュール構造グラフ内のすべてのノードを含む順序セットａｒｒ２を作成するステップと、

（ｉｉｉ）ａｒｒ１をたどり、ビヘイビア構造グラフ内の基底変数を識別するステップと、

（ｉｖ）ビヘイビア構造グラフ内の非基底変数を基底変数で表すステップと、

（ｖ）スケジュール構造グラフ内の入力ノードに対する等価性リストを構成するステップと、

（ｖｉ）ａｒｒ２をたどり、ａｒｒ２内の各ノードを処理して、スケジュール構造グラフの入力からスケジュール構造グラフの出力へ等価性リストを伝搬させるステップと、

（ｖｉｉ）各等価性リスト内のエントリは対（ｕ，ｃ）であり、ｕはビヘイビア構造グラフ内の信号の識別子であり、ｃは等価性の条件を表す二分決定ダイヤグラムであるとして、ビヘイビア構造グラフ内の対応する出力ノードで等価性が確定したかどうか、及び、対応する条件ｃがａｒｒ２内のプライマリ出力ノードに対するトートロジーであるかどうかをチェックするステップと、

（ｖｉｉｉ）ａｒｒ２内のすべての出力ノードについて前記ステップ（ｖｉｉ）を繰り返すステップと、

（ｉｘ）すべての出力ノードが等価であることがわかった場合に等価性を見つけたとするステップと、

を用いて前記ステップ（ｆ）を実行することを可能にする命令をさらに含むことを特徴とする請求項２８に記載のコンピュータシステム。

【請求項３０】　回路のスケジュールと該回路のビヘイビアとの間の等価性を検証するための、プロセッサ及びメモリを有するコンピュータシステムにおいて、

前記スケジュール及び前記ビヘイビアは、実行の巡回スレッドを有する可能性があり、

前記メモリは、前記コンピュータシステムが、

（ａ）スケジュールをスケジュール状態遷移グラフとして表現するステップと、

（ｂ）ビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、

（ｃ）前記スケジュール状態遷移グラフ内の強連結成分を識別するステップと、

（ｄ）各強連結成分内の終了ノードを識別するステップと、

（ｅ）前記スケジュール状態遷移グラフをつぶして、前記強連結成分を通らないサブパスを併合するステップと、

（ｆ）以前に選択されていないパスを選択するステップと、

（ｇ）前記ステップ（ｆ）で選択されたパスに対する構造ＲＴＬ回路を取得するステップと、

（ｈ）選択されたパスを列挙するのに必要なすべての状態遷移決定をカプセル化するパスシグナルを生成するための回路を構造ＲＴＬ回路に追加するステップと、

（ｉ）パスシグナルを用いて、制約されたシンボリックシミュレーションを実行してビヘイビア状態遷移グラフ内の対応するパスを識別するステップと、

（ｊ）選択されたパスにおいて、以前に選択されていない強連結成分を選択するステップと、

（ｋ）選択されたパス内の選択された強連結成分に対する不変項を、対応セットのリストとして抽出するステップと、

（ｌ）対応セットのリストから１つの対応セットを選択するステップと、

（ｍ）選択された対応セットが、前のシンボリックシミュレーションの強連結成分カットにおいて得られる変数対応より小さい場合に、シンボリックシミュレーションを再実行するステップと、

（ｎ）対応セットのリスト内の各対応セットについて前記ステップ（ｉ）～（ｍ）を繰り返すステップと、

（ｏ）出力等価性条件が、非等価性を報告するパス条件以外の条件付きであるかどうかをテストするステップと、
（ｐ）前記ステップ（ｏ）で前記出力等価性が条件付きである場合にこの検証を終了するステップと、
（ｑ）選択されたパス内のすべての強連結成分について前記ステップ（ｊ）～（ｐ）を繰り返すステップと、
（ｒ）終了点が高々３度現れるようにルートからシンクへのすべてのパスについて前記ステップ（ｆ）～（ｑ）を繰り返すステップとを用いて前記検証を実行することを可能にすることを特徴とする、回路のスケジュールと該回路のビヘイビアとの間の等価性を検証するためのコンピュータシステム。
【請求項３１】　前記命令は、前記コンピュータシステムが、
（ｉ）ビヘイビア状態遷移グラフの始状態を許容パスリストに割り当てるステップと、
（ｉｉ）許容パスリスト内で以前に訪れていない状態を選択するステップと、
（ｉｉｉ）ビヘイビア構造ＲＴＬを生成するステップと、
（ｉｖ）非解釈シンボリックシミュレーションを実行して、スケジュール構造ＲＴＬ及びビヘイビア構造ＲＴＬ内の対応する信号を識別するステップと、
（ｖ）遷移条件とパスシグナルの論理積がゼロでない場合に、状態S_jの新しいコピーを許容パスに追加するステップと、
（ｖｉ）S_iからS_jへの各出遷移ごとに前記ステップ（ｖ）を繰り返すステップと、
（ｖｉｉ）許容パス内に残る訪れていない状態のみが終状態のインスタンスとなるまで、すべての訪れていない状態について前記ステップ（ｉｉｉ）～（ｖｉ）を繰り返すステップとを用いてステップ（ｉ）を実行することを可能にする命令をさらに含むことを特徴とする請求項３０に記載のコンピュータシステム。
【請求項３２】　前記命令は、前記コンピュータシステムが、各ループごとに、
（ｉ）各カットが前記ループの各実行の境界における変数値を表すような、スケジュール内のパスの構造ＲＴＬ回路内の３個のカットを識別するステップと、
（ｉｉ）ビヘイビアにおけるパスの構造ＲＴＬ回路内の対応するカットを識別して、第１と第２のカットの間のサブ回路と、第２と第３のカットの間のサブ回路が同型であることをチェックするステップと、
（ｉｉｉ）スケジュール及びビヘイビアのＲＴＬ回路における対応するカットの各対における変数どうしの間の等価関係を識別するステップと、
（ｉｖ）最後のカットと最後の前のカットとの間の等価関係が同一であるかどうかをチェックするステップと、
（ｖ）前記ステップ（ｉｖ）の関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットである場合、最後の前のカットにおける等価関係を破棄し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、前記ステップ（ｉｉｉ）から繰り返すステップと、
（ｖｉ）前記ステップ（ｉｖ）の関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットでない場合、最後の前のカットにおける等価関係を、等価関係セットの集合に追加し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、前記ステップ（ｉｉｉ）から繰り返すステップと、
（ｖｉｉ）前記ステップ（ｉｖ）の関係が同一である場合、最後のカットにおける等価関係を、等価関係セットの集合に追加するステップと、
（ｖｉｉｉ）等価関係セットの集合内で、他のエントリのスーパーセットであるすべてのエントリを削除するステップと、
（ｉｘ）等価関係セットの最終集合を、不変項の所望の集合として指定するステップと、
を用いて前記ステップ（ｋ）を実行することを可能にする命令をさらに含むことを特徴とする請求項３０記載のコンピュータシステム。
【請求項３３】　コンピュータが回路のスケジューリングの正当性をチェックすることを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品において、
回路に対するスケジュールは、ビヘイビア記述から得られ、
前記コンピュータコードは、
ループが存在するときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出するコンピュータコードと、
ループ不変項を抽出するためのシンボリックシミュレーションのコンピュータコードと、
非巡回スレッドの等価性を証明するコンピュータコードとを含むことを特徴とする、コンピュータが回路のスケジューリングの正当性をチェックすることを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品。
【請求項３４】　前記ビヘイビア記述は、サイクル境界の導入によって変換されることを特徴とする請求項３３記載のコンピュータプログラム製品。
【請求項３５】　前記ビヘイビア記述は、演算並べ替えによって変換されることを特徴とする請求項３３記載のコンピュータプログラム製品。
【請求項３６】　前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換されることを特徴とする請求項３３記載のコンピュータプログラム製品。

11

【請求項３７】　前記ビヘイビア記述は、演算の投機実行によって変換されることを特徴とする請求項３３記載のコンピュータプログラム製品。

【請求項３８】　コンピュータが回路のビヘイビア記述に対して回路のスケジュールを検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品において、該コンピュータコードは、

前記コンピュータが、スケジュールをスケジュール状態遷移グラフとして指定することを可能にするスケジュール状態遷移グラフジェネレータコードと、

前記コンピュータが、回路のビヘイビアをビヘイビア状態遷移グラフとして指定することを可能にするビヘイビア状態遷移グラフジェネレータコードと、

前記コンピュータが、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択することを可能にするスケジュールスレッドセレクタコードと、

前記コンピュータが、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを選択することを可能にするビヘイビアスレッドセレクタコードと、前記コンピュータが、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換することを可能にするコンバータコードと、

前記コンピュータが、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックすることを可能にする等価性チェッカコードと、

からなることを特徴とする、コンピュータが回路のビヘイビア記述に対して回路のスケジュールを検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品。

【請求項３９】　コンピュータが回路のビヘイビア記述に対して回路のスケジュールを検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品において、

前記コンピュータコードは、前記コンピュータが、

（ａ）スケジュールをスケジュール状態遷移グラフとして指定するステップと、

（ｂ）回路のビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、

（ｃ）前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するステップと、

（ｄ）前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを識別するステップと、

（ｅ）前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、

（ｆ）前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、

（ｇ）実行のすべてのスレッドについて前記ステップ

12

（ｃ）～（ｆ）を繰り返すステップと、

を実行することを可能にすることを特徴とする、コンピュータが回路のビヘイビア記述に対して回路のスケジュールを検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品。

【請求項４０】　前記コンピュータコードは、前記コンピュータが、

（ｉ）前記ビヘイビア状態遷移グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記ビヘイビア構造グラフ内のすべてのノードを含む順序セットａｒｒ１を作成するステップと、

（ｉｉ）前記ビヘイビア構造グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記スケジュール構造グラフ内のすべてのノードを含む順序セットａｒｒ２を作成するステップと、

（ｉｉｉ）ａｒｒ１をたどり、ビヘイビア構造グラフ内の基底変数を識別するステップと、

（ｉｖ）ビヘイビア構造グラフ内の非基底変数を基底変数で表すステップと、

（ｖ）スケジュール構造グラフ内の入力ノードに対する等価性リストを構成するステップと、

（ｖｉ）ａｒｒ２をたどり、ａｒｒ２内の各ノードを処理して、スケジュール構造グラフの入力からスケジュール構造グラフの出力へ等価性リストを伝搬させるステップと、

（ｖｉｉ）各等価性リスト内のエントリは対（ｕ，ｃ）であり、ｕはビヘイビア構造グラフ内の信号の識別子であり、ｃは等価性の条件を表す二分決定ダイヤグラムであるとして、ビヘイビア構造グラフ内の対応する出力ノードで等価性が確定したかどうか、及び、対応する条件ｃがａｒｒ２内のプライマリ出力ノードに対するトートロジーであるかどうかをチェックするステップと、

（ｖｉｉｉ）ａｒｒ２内のすべての出力ノードについて前記ステップ（ｖｉｉ）を繰り返すステップと、

（ｉｘ）すべての出力ノードが等価であることがわかった場合に等価性を見つけたとするステップと、

を用いて前記ステップ（ｆ）を実行することを可能にすることを特徴とする請求項３９記載のコンピュータプログラム製品。

【請求項４１】　コンピュータが回路のスケジュールと該回路のビヘイビアとの間の等価性を検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品において、

前記スケジュール及び前記ビヘイビアは、実行の巡回スレッドを有する可能性があり、

前記コンピュータコードは、前記コンピュータが、

（ａ）スケジュールをスケジュール状態遷移グラフとし

て表現するステップと、
（ｂ）ビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、
（ｃ）前記スケジュール状態遷移グラフ内の強連結成分を識別するステップと、
（ｄ）各強連結成分内の終了ノードを識別するステップと、
（ｅ）前記スケジュール状態遷移グラフをつぶして、前記強連結成分を通らないサブパスを併合するステップと、
（ｆ）以前に選択されていないパスを選択するステップと、
（ｇ）前記ステップ（ｆ）で選択されたパスに対する構造ＲＴＬ回路を取得するステップと、
（ｈ）選択されたパスを列挙するのに必要なすべての状態遷移決定をカプセル化するパスシグナルを生成するための回路を構造ＲＴＬ回路に追加するステップと、
（ｉ）パスシグナルを用いて、制約されたシンボリックシミュレーションを実行してビヘイビア状態遷移グラフ内の対応するパスを識別し、該パスに対する構造ＲＴＬ回路を取得するステップと、
（ｊ）選択されたパスにおいて、以前に選択されていない強連結成分を選択するステップと、
（ｋ）選択されたパス内の選択された強連結成分に対する不変項を、対応セットのリストとして抽出するステップと、
（ｌ）対応セットのリストから１つの対応セットを選択するステップと、
（ｍ）選択された対応セットが、前のシンボリックシミュレーションの強連結成分カットにおいて得られる変数対応より小さい場合に、シンボリックシミュレーションを再実行するステップと、
（ｎ）対応セットのリスト内の各対応セットについて前記ステップ（ｉ）～（ｍ）を繰り返すステップと、
（ｏ）出力等価性条件が、パス条件以外の条件付きであるかどうかをテストするステップと、
（ｐ）前記ステップ（ｏ）で前記出力等価性が条件付きである場合に非等価性を報告してこの方法を終了するステップと、
（ｑ）選択されたパス内のすべての強連結成分について前記ステップ（ｊ）～（ｐ）を繰り返すステップと、
（ｒ）終了点が高々３度現れるようにルートからシンクへのすべてのパスについて前記ステップ（ｆ）～（ｑ）を繰り返すステップと、
を実行することを可能にすることを特徴とする、コンピュータが回路のスケジュールと該回路のビヘイビアとの間の等価性を検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品。
【請求項４２】　前記コンピュータコードは、前記コンピュータが、
（ｉ）ビヘイビア状態遷移グラフの始状態を許容パスリストに割り当てるステップと、
（ｉｉ）許容パスリスト内で以前に訪れていない状態を選択するステップと、
（ｉｉｉ）ビヘイビア構造ＲＴＬを生成するステップと、
（ｉｖ）非解釈シンボリックシミュレーションを実行して、スケジュール構造ＲＴＬ及びビヘイビア構造ＲＴＬ内の対応する信号を識別するステップと、
（ｖ）遷移条件とパスシグナルの論理積がゼロでない場合に、状態S_jの新しいコピーを許容パスに追加するステップと、
（ｖｉ）S_iからS_jへの各出遷移ごとに前記ステップ（ｖ）を繰り返すステップと、
（ｖｉｉ）許容パス内に残る訪れていない状態のみが終状態のインスタンスとなるまで、すべての訪れていない状態について前記ステップ（ｉｉｉ）～（ｖｉ）を繰り返すステップと、
を用いて、前記ステップ（ｉ）の制約されないシンボリックシミュレーションを実行することを可能にすることを特徴とする請求項４１記載のコンピュータプログラム製品。
【請求項４３】　前記コンピュータコードは、前記コンピュータが、各ループごとに、
（ｉ）各カットが前記ループの各実行の境界における変数値を表すような、スケジュール内のパスの構造ＲＴＬ回路内の３個のカットを識別するステップと、
（ｉｉ）ビヘイビアにおけるパスの構造ＲＴＬ回路内の対応するカットを識別して、第１と第２のカットの間のサブ回路と、第２と第３のカットの間のサブ回路が同型であることをチェックするステップと、
（ｉｉｉ）スケジュール及びビヘイビアのＲＴＬ回路における対応するカットの各対における変数どうしの間の等価関係を識別するステップと、
（ｉｖ）最後のカットと最後の前のカットとの間の等価関係が同一であるかどうかをチェックするステップと、
（ｖ）前記ステップ（ｉｖ）の関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットである場合、最後の前のカットにおける等価関係を破棄し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、前記ステップ（ｉｉｉ）から繰り返すステップと、
（ｖｉ）前記ステップ（ｉｖ）の関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットでない場合、最後の前のカットにおける等価関係を、等価関係セットの集合に追加し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、前記ステップ（ｉｉｉ）から繰り返すステップと、

（ｖｉｉ）前記ステップ（ｉｖ）の関係が同一である場合、最後のカットにおける等価関係を、等価関係セットの集合に追加するステップと、

（ｖｉｉｉ）等価関係セットの集合内で、他のエントリのスーパーセットであるすべてのエントリを削除するステップと、

（ｉｘ）等価関係セットの最終集合を、不変項の所望の集合として指定するステップと、

を用いて前記ステップ（ｋ）で不変項を抽出することを可能にすることを特徴とする請求項４１記載のコンピュータプログラム製品。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】［１．発明の詳細な説明］

［１．１　発明の属する技術分野］本発明はハイレベル合成におけるスケジューリングステップの検証(verification)に関する。本発明の主要な焦点は、スケジューリングとともに実行される可能性が高いすべての代表的な変換を含むスケジューリング検証のための新規技術にある。特に、本発明は、ループと、スケジューリング中に実行されるさまざまなループ変換を扱うことが可能な検証技術を提供する。

【０００２】

【従来の技術】［１．２　従来の技術］回路を出荷するまでの時間を短縮する手段として、ハイレベル仕様からの合成が重要であることはよく認識されている。これにより、高速合成が可能になるのに加えて、再使用の観点からもより有利となる。Chrysalis(R)、Synopsys(R)及びその他の多くの会社から提供されるツールを用いた組合せ論理の検証は、初期ネットリスト仕様に対して最終論理ネットリストの妥当性の検証を行う必要があるために、ハイレベル・ビヘイビア（動作）記述から得られるレジスタトランスファレベル（ＲＴＬ）のネットリストを検証するためのツールも必要となる。本発明は、検証を実行する技術を改善するためのものである。よく知られているように、シミュレーションは、正当性(correctness)を保証しないにも拘わらず時間がかかるために、検証ストラテジとして十分ではあり得ない。そこで、フォーマル検証の方法論が必要となる。

【０００３】初期ビヘイビア仕様から最終ＲＴＬを実現するために適用される変換のスコープ（有効範囲）が与えられている場合、入力として単に２つの大幅に異なるレベルでの記述をとるブラックボックス検証システムは、すべての実際的な目的で実現可能なわけではない。幸いに、合成自体は、自動ツールを用いてなされるか手動でなされるかにかかわらず、一般的に、スケジューリング、リソース割当て及びレジスタ代入のような明確に区分された基本的なステップからなる共通の基本フローに従う。検証方法が実際的であるためには、このフローの知識を活用しなければならない。実際、スケジューリングやレジスタ代入のようなステップどうしの間の区分をそのまま保持することは、「検証のための設計」の良いストラテジである。最終設計の品質が多少犠牲になったとしても、合成プロセスははるかに検証容易になる。

【０００４】ハイレベル合成フロー中の個々のステップの検証は、合成プロセス全体を検証するよりは容易であるものの、決して簡単ではない。スケジューリングとは、タイムスタンプを演算(operation)に割り当てる作業である。同期設計では、これは、演算に状態を対応させることによって行われる。さまざまな設計要件を満たすために、演算並べ替え、ループ展開、投機実行（speculative execution）等のような変換が、このステップ中に実行されることがある。スケジューリングをチェックしようとする検証ツールにとって最小限の要件は、これらの変換をそのスコープ（有効範囲）に含むことである。

【０００５】本明細書において、シンボリックシミュレーションとは、回路を通して、変数値ではなく、変数を伝搬させる手続きを含意する。「非解釈(uninterpreted)」という用語は、この場合、標準の算術演算のような複雑な演算に遭遇したときに、入力のブール演算の値ではなく、入力リスト及び演算名が転送されることを意味する。

【０００６】１．２．１　関連する研究

従来、ハイレベル記述から生成される設計を検証するためのいくつかの技術が提案されている。プログラム及びハードウェアの検証のためのシンボリックシミュレーションに関するかなりの研究活動が７０年代及び８０年代になされた。代表的なものとして、J. Darringer, "The application of program verificationtechniques to hardware verification", in Proc. Design Automation Conf., pp. 375-381, June 1979、を参照。しかし、Darringerの研究及びそこから派生した研究は、スケジューリングを検証するという場合に応用を限定している。派生した研究の一部は、

・W. Cory, "Symbolic simulation for functional verification with ADLIBand SDL", in Proc. Design Automation Conf., pp. 82-89, June 1981

・V. Pitchumani and E. Stabler, "A formal method for computer design verification", in Proc. Design Automation Conf., pp. 809-814, June 1982

に見られる。

【０００７】重要な点として、Darringerの研究の主要な制限は、シンボリックシミュレータがチェックを実行するための不変項(invariant)を設けることをユーザに要求していたことである。実際、知られているように、２つのハードウェア記述を比較する際に、不変項は、一方の記述の完全な状態が他方の状態と一致しなければならない対応点（Darringerの用語では制御点(control point)）である。スケジューリングの場合、シミュレータ

にこの情報を提供するために、ユーザは、例えば、合成ツールによって実行されるループ変換の詳細な知識を有する必要がある。このような要求は困難である。さらに、このような要求は、検証の目的に部分的に反することになる。また、ユーザが対応点を提供する場合、完全性の問題は未解決のままとなる。制御点どうしの間の中間信号間の対応を検出し、それを利用して、制御点における同型(isomorphism)のためにチェックすべき式を単純化する追加能力を有する同じ基本的なアルゴリズムが、C.-T. Chen and A. Parker, "A hybrid numeric/symbolic program for checking functional and timingcompatibility of synthesized design", in Proc. The International Symposium on High Level Synthesis, pp.112-117, May 1994、で提案された。

【０００８】他のいくつかの関連する文献についてもここで説明する。Minatoは、２つのハードウェア記述どうしの間の等価性を確かめるためのＢＤＤ（Binary DecisionDiagram：二分決定ダイヤグラム）に基づくアプローチを提案している。S. Minato, "Generation of BDDs from hardware algorithm descriptions", in Proc. Int. Conf. Computer-Aided Design, pp.644-649, Nov. 1996、を参照。このアプローチでは、すべての条件分岐は、追加変数の使用により直線的なコードに変換される。さらに、ループは、すべての変数に対するＢＤＤが追加展開で変化しなくなるまで各ループを展開することによって処理される。この方法は、算術関数を表現する際のＢＤＤの制限と、ループ終了条件が満たされるまでループを明示的に展開する必要があることとによる欠点がある。Gong et al.は、ハイレベル合成におけるさまざまなステップをチェックするための規則スイートのセットを提案した。J. Gong, C. T. Chen, and K. Kucukcakar, "Multi-dimensional rule checking for high-level design verification", in Proc. Int. High-level Design Validation & Test Wkshp., Nov. 1997、を参照。しかし、彼らの等価性チェッカは、構造同型をチェックすることに制限されていた。Bergamaschiand Rajeの貢献は、２つの記述における対応する信号が相異なる時点で観測されなければならないときにどのようにすれば等価性チェックを実行することができるかを示したことである。R. A. Bergamaschi and S. Raje, "Observable timewindows: Verifying high-level synthesis results", IEEE Design & Test ofComputers, vol.8, pp.40-50, Apr. 1997、を参照。

【０００９】最近では、検証において算術及び制御算術相互作用をモデル化するためのいくつかの技術が提案されている。

・K. T. Cheng and A. S. Krishnakumar, "Automatic functional test generation using the extended finite state machine model", in Proc. Design Automation Conf., June 1993

・F. Fallar, S. Devadas, and K. Keutzer, "Functional vector generationfor HDL models using linear programming and 3-satisfiability", in Proc.Design Automation Conf., June 1998

・J. Kukula, T. Shiple, and A. Aziz, "Implicit state enumeration for FSMs with datapaths", in Proc. Formal Methods in Computer Aided Design, Nov. 1998

を参照。これらの技術は強力であり、ハイレベル合成から生成される設計の検証におけるモデルチェック技術や定理証明とともに、将来の応用の可能性がある。

・J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill, "Symbolic model checking for sequential circuit verification", IEEE Transactions on Computer-Aided Design, vol.13, Apr. 1994

・R. K. Brayton et al., "VIS: A system for verification and synthesis", in Proc. Int. Conf. Computer-Aided Verification, July 1996

・S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system", in 11th International Conference on Automated Deduction (D. Kapur, ed.), vol.607 of Lecture Notes in Artificial Intelligence, Springer Verlag, 1992

【００１０】ここ数年、非解釈関数によるシンボリックシミュレーションに基づく等価性チェックのための基本的アルゴリズムの効率の改善について、いくつかの論文が発表されている。

・R. Shostak, "An algorithm for reasoning about equality", Communications of the ACM, vol.21, no.7, pp.583-585, 1978

・R. Jones, D. Dill, and J. Burch, "Efficient validity checking for processor validation", in Proc. Int. Conf. Computer-Aided Design, pp.2-6,Nov. 1995

・A. Goel, K. Sajid, H. Thou, A. Aziz, and V. Singhal, "BDD based procedures for a theory of equality with uninterpreted functions", in Proc.Int. Conf. Computer-Aided Verification, pp.244-255, July 1998

を参照。本発明で用いられるシンボリックシミュレーションアルゴリズムは、従来技術といくつかの共通点を有する。その決定手続きは、算術演算とともに、ブール演算を含む。

・C. Barrett, D. Dill, and J. Levitt, "Validity checking for combinations of theories with equality", in Proc. Formal Methods in Computer Aided Design, pp.187-201, Nov. 1996

・A. Goel, K. Sajid, H. Thou, A. Aziz, and V. Singhal, "BDD based procedures for a theory of equality with uninterpreted functions", in Proc.Int. Conf. Computer-Aided Verification, pp.244-255, July 1998

を参照。また、要求に応じて、決定手続きに追加の代数を加えることも可能である。C. Barrett, D. Dill, and J. Levitt, "Validity checking for combinations of theories with equality", in Proc. Formal Methods in Computer Aided Design, pp. 187-201, Nov. 1996、を参照。

【００１１】しかしながら、本発明で用いられるシンボリックシミュレーションアルゴリズムは、ブール演算／条件をどのように扱うかにおいて従来技術とは異なる。最も近いのはA. Goel et al.のものであるが、対応する信号を記憶するのに必要なブックキーピングにおいて異なる。A. Goel, K. Sajid, H. Thou, A. Aziz, and V. Singhal, "BDD based procedures for a theory of equality with uninterpreted functions", in Proc. Int. Conf. Computer-Aided Verification, pp. 244-255, July 1998、を参照。

【００１２】１．２．２　従来技術：スケジューリングの有効範囲

スケジューリングは、ハイレベル合成に基づく設計フローにおいて最も重要なステップのうちの１つである。スケジューリングに関する全般的な情報については、

・D. D. Gajski, N. D. Dutt, A. C.-H. Wu, and S. Y.-L. Lin, High-levelSynthesis: Introduction to Chip and System Design, Kluwer Academic Publishers, Norwell, MA, 1992

・G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York, NY, 1994

を参照。タイミング情報を部分的にしか又は全く含まないビヘイビア記述からはじめて、設計のサイクルごとのビヘイビアは、スケジューリングステップ中に固定される。このサブセクションでは、スケジューリングステップ中に実行されるいくつかの代表的な変換について説明する。それらの変換により検証プロセスの複雑さがどのように増大するかについてもここで説明する。

【００１３】［１．２．２．１　クロックサイクル境界の導入］スケジューリングは、回路のビヘイビア記述からスケジュールを導出するプロセスである。単純な形のスケジューリングでは、実行されるのは、ビヘイビア記述にクロックサイクル境界、すなわちカットを入れることからなる変換だけである。ＨＤＬ記述の場合、これに相当する可能性のあるものの１つは、ビヘイビア記述にいくつかの"wait until clk=1 and clk"イベント文を挿入することである。詳細は、D. Knapp, T. Ly, D. MacMillen, and R. Miller, "Behavioral synthesis methodology for HDL-based specification and validation", in Proc. Design Automation Conf., pp. 28-291, June 1995、を参照。あるサイクル境界と次のサイクル境界の間の演算の列は組合せ論理を表すため、一般に、いくつかの条件を満たすために複数のカットを入れる。例えば、すべてのループ（ＶＨＤＬのprocess文やVerilogのalwaysブロックのような暗黙のループを含む）を切るためにカットを入れる。知られているように、ビヘイビアとスケジュールとは、サイクルごとに等価ではない。従って、等価性の概念と、等価性をチェックする技術とは、クロックサイクル境界を超えて作用する必要がある。周知のように、出力を計算するのに必要なクロックサイクル数は、異なるスレッドあるいは入力値に対しては異なる可能性がある。さらに、（データ依存性がある可能性もある）ループの存在もまた、検証の複雑さを増大させる。さらに、ＨＤＬの複雑なセマンティクス（例えば、信号代入や並行文）のため、サイクル境界を導入するという単純な変換でさえ、設計の機能を変更することがある。これは、次の例によって明確に例示される。

【００１４】例１：図１に示すＶＨＤＬ^{TM}記述を考える。この記述は、whileループと、さまざまな変数及び信号代入文を含むプロセスに関する。いくつかのステップは、算術計算を含む。このプロセスでは、２つの"wait until clk='1' and clk'"イベント文に注釈を付けてある。これらのイベント文は、スケジューリング中に追加されたクロックサイクル境界を示す。なお、x#var、y#var、u#var及びdx#varは信号であり、これらの変数に対してなされるすべての代入は信号代入である。ＶＨＤＬにおける信号代入文のセマンティクスは、信号に代入される値は即時に計算されるが、その代入はある後の時刻まで有効にならないというものである。この時刻は、明示的な時刻が指定されていない場合、デフォルトでは、デルタに等しい。"wait for 0ns;"文の目的は、デルタ遅延を導入し、先行する信号代入文によって生成された新しい値が有効になることを強制することである。

【００１５】ビヘイビア記述におけるwhileループ内の信号y#varへの代入を考える（なお、このビヘイビア記述は、"wait until clk='1' and clk'event"文を含まない）。右辺の式の計算は、信号u#varの古い値を使用する。信号u#varへの先行する代入は実行されているが、ループの最後の"wait for 0ns"文まで有効ではないからである。しかし、スケジュールでは、u#varへの信号代入の後に"wait until clk='1' and clk'event"文を導入することにより、y#varへの代入が評価される前にu#varの新しい値が有効になることが強制される。上記の差の結果として、スケジュールは、シミュレーション中に誤った値を生成する可能性がある。

【００１６】［１．２．２．２　演算の並べ替え］演算の並べ替えは、ビヘイビア記述に存在する並列性を利用するため、及び、与えられたリソースを最大限に利用するために、スケジューリング中に実行することが可能である。一般に、これは条件演算及び完全なループを並べ替えることを含む。最新のスケジューリング技術では、しばしば、データフロー及びメモリアクセスの依存性を維持しながら、ビヘイビア記述における演算を任意に並べ替える。詳細には、

・D. D. Gajski, N. D. Dutt, A. C.-H. Wu, and S. Y.-L. Lin, High-levelSynthesis: Introduction to Chip and System Design, Kluwer Academic Publishers, Norwell, MA, 1992

・G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York, NY, 1994

を参照。演算の並べ替え中に導入される可能性のあるエラーには、データ依存性、条件制御依存性、及びメモリハザード（例えば、ＲＡＷ(read-after-write)、ＷＡＷ(write-after-write)など）の違反がある。演算の並べ替えにより生成されるこのようなスケジュールの検証は、スケジュールからの制御及びデータのフローの抽出を必要とする。さらに、制御及びデータの依存性が実装において満たされることをチェックすること（例えば、構造同型チェック技術や規則チェック技術を用いて）が含まれる。詳細には、J. Gong, C. T. Chen, and K. Kucukcakar, "Multi-dimensional rule checking for high-level design verification", in Proc. Int. High-level Design Validation & Test Wkshp., Nov. 1997、を参照。

【００１７】例２：図２（ａ）はビヘイビアＣ記述を示し、図２（ｂ）はその対応するスケジュールを示す。この例のビヘイビアは、シーケンシャルプログラムとして指定されているため、各スレッドで実行される演算について完全な順序を定義している。しかし、スケジューラは、保存の必要がある演算どうし間の依存性の解析を自動的に実行するかもしれないし、演算の順序が出力の計算にとって重要ではないときに演算を並べ替えることを選択することがある。このような並べ替えは、リソースやクロック期間の数を最適にするために実行される可能性もある。

【００１８】以下の並べ替え操作が、このビヘイビアについてのスケジュールで実行されている。

・ビヘイビアにおいて＋２及び＊１とマークされた演算の順序は逆転されている。これは、基本ブロック内の演算の局所並べ替えの例である。この並べ替えは正しくない。その理由は、ビヘイビアにおける演算＋２と＊１の間にデータ依存性があり（＋２の出力は＊１の入力である）、このデータ依存性は、図２に示すスケジュールでは破られているからである。

【００１９】２つのforループの実行順序はスケジュールでは逆転されている。ビヘイビアにおいて最初に現れるループは、スケジュールでは状態Ｓ２、Ｓ３、及びＳ４によって実現され、ビヘイビア記述の第２のforループは、スケジュールの状態Ｓ１で実現されている。この並べ替えは妥当である。その理由は、２つのループの間にデータ依存性や優先順位制約がないからである（これらのループに共通な唯一の変数であるループカウンタcountlは、各ループの前に０に初期化される）。

【００２０】［１．２．２．３　パス／セグメントの複製］ビヘイビア記述における相異なるパス（すなわち、計算のスレッド）は、しばしば、異なるスケジューリングの機会及び制約を提示する。従って、ビヘイビアにおいて与えられたパスを最大限に最適化するためには、ビヘイビアにおける残りのパスとは別個にパス（またはその部分）をスケジューリングする必要が生じることがある。これにより、スケジュールにおいてパスまたはパスセグメントの複製が生じる。パスに基づくスケジューリング技術は、ビヘイビアにおける単純（無閉路的あるいは非巡回（acyclic））パスに対してこのような最適化を行う。同様に、ループ指向スケジューリング技術は、ビヘイビアにおける非単純パスに対してこのような最適化を自動的に行う。

・R. Camposano, "Path-based scheduling for synthesis", IEEE Trans. Computer-Aided Design, vol.10, pp.85-93, Jan. 1991

・S. Bhattacharya, S. Dey, and F. Brglez, "Performance analysis and optimization of schedules for conditional and loop-intensive specifications", in Proc. Design Automation Conf., pp.491-496, June 1994

を参照。

【００２１】また、スケジューリング中のパス／セグメントの複製は、検証プロセスの複雑さを増大させる。知られているように、演算と変数の間の関係は一対一ではなくなる。従って、構造同型をチェックする単純な技術は、スケジュールとビヘイビアの等価性を証明するのに十分ではない。複製により、ビヘイビアに対するスケジュールにおける演算の数が増大するが、ビヘイビア、あるいは、そのビヘイビア内の与えられたパスあるいはスレッドに沿って実行される演算のセットは同一である。このように、従来の検証ストラテジは、ビヘイビア及びスケジュールにおけるパスを列挙することである。さらに、対応するパスのそれぞれの対ごとに、このようなストラテジは、ビヘイビア及びスケジュールにおいて実行される演算のセットが同型のデータフローグラフを形成することを検証する。さらに詳細には、C.-T. Chen and A. Parker, "A hybrid numeric/symbolic program for checking functional and timing compatibility of synthesized designs", inProc. The International Symposium on High-Level Synthesis, pp.112-117, May 1994、を参照。

【００２２】［１．２．２．４　ループ変換］ループは、しばしば、ビヘイビア記述においてパフォーマンスあるいはパワーに関するクリティカルな部分を構成する。データ独立ループ（実行回数が事前に既知であり、入力値とは独立なループ）、及び、データ依存ループ（実行回数が静的に既知ではなく、入力データに依存するループ）を積極的に最適化するさまざまなスケジューリング技術が提案されている。これらの技術には以下の

ものがある。

【００２３】・ループ展開。ループ展開の１つの意味は、ビヘイビアにおけるループをループ本体(loop body)のいくつかのコピーに変換した後、そのループのコピーをすることである。第２の意味は、スケジュールにおけるループの１回の実行が、ビヘイビアにおけるループの複数回の実行に対応することである。２種類のループ展開変換を図３（ｂ）及び図３（ｃ）に例示する。

【００２４】・ループ回転。これにより、スケジュールにおけるループの境界は、ビヘイビアにおける対応するループの境界に対してずれる。ループ回転を図３（ｄ）に例示する。

【００２５】・ループパイプライン化。これは、ループ折畳み(loop folding)あるいはループ巻付け(loop winding)ともいい、ループ本体の複数回の実行を並行して実行するものである。これには、正当性を保証するためにプロローグ及びエピローグを作成することも必要になることがある。さらに詳細には、R. Potasman, J. Lis, A. Nicolau, and D. Gajski, "Percolation based synthesis", in Proc. Design Automation Conf., pp. 444-449, June 1990、を参照。ループパイプライン化を図３（ｅ）に例示する。

【００２６】ビヘイビアにおけるループの存在と、スケジューリング中のループ最適化の適用は、検証を非常に複雑にする。特に、ビヘイビア及びスケジュールにおけるスレッドあるいはパスの列挙は、ループの相異なる実行カウントを考慮する必要がある。さらに、ループが実行される回数はデータ依存であることがあり、静的に限定することが困難である。さらに、このような限定が可能である場合、あるいは、ループ実行回数が一定で既知である場合であっても、ビヘイビア及びスケジュールにおける異なるパスの個数により、すべてのこのようなパスの列挙は至難となる。さらに、回転やパイプライン化のようなループ最適化は、スケジュールとビヘイビアにおけるループの境界どうしの間の対応を破壊する。本発明の重要な特徴は、スケジュールにおけるすべての非単純パスの列挙を避ける、ループ不変項の自動抽出にある。

【００２７】［１．２．２．５　投機実行］投機実行では、ビヘイビア記述の一部が、その部分を実行することが必要であるとわかる前に、実行される。投機実行は、ハイレベル合成のスケジューリングステップに統合されると、大幅なパフォーマンス改善が得られる。しかし、投機実行によって、検証は更に複雑になる。重要な点であるが、ビヘイビアでの制御依存性は、投機実行を含むスケジュールでは満たされない。さらに詳細には、

・I. Radivojevic and F. Brewer, "Ensemble representation and techniques for exact control-dependent scheduling", in Proc. High-level SynthesisWorkshop, pp. 60-65, 1994

・O. Lakshminarayana, A. Raghunathan, and N. K. Jha, "Incorporating speculative execution into scheduling for control-flow intensive behaviors", in Proc. Design Automation Conf., pp. 108-113, June 1998

を参照。

【００２８】スケジューラは、投機実行される演算の結果を格納するためにスケジュールに追加一時変数を導入するのが一般的である。また、スケジューラは、それらの一時変数が依存する投機条件が評価された後にそれらの一時変数を解決するための追加コード（代入文）を生成する。構造同型に基づく検証技術は、このような変換を検証することができない。これについては、

・J. Gong, C. T. Chen, and K. Kucukcakar, "Multi-dimensional rule checking for high-level design verification", in Proc. Int. High-level Design Validation & Test Wkshp., Nov. 1997

・C.-T. Chen and A. Parker, "A hybrid numeric/symbolic program for checking functional and timing compatibility of synthesized design", in Proc. The International Symposium on High Level Synthesis, pp. 112-117, May1994

に説明されている。

【００２９】

【発明が解決しようとする課題】［２．発明の概要］本発明は、新規な非解釈シンボリックシミュレーション手続きに関する。本発明の技術は、ビヘイビア仕様及びスケジューリングされたＲＴＬが与えられた場合に、２つの記述の出力が相互に無条件に対応するかどうかを判定する。

【００３０】スケジューリングされたＲＴＬとビヘイビア記述の間の、条件付きの可能性がある入力対応のリストからはじめて、本発明の技術は、２つの記述における信号の間の条件付き信号対応を出力へ向かって伝搬させる。２つの演算の出力は、その演算型が同一であり、かつ、ある条件下でその演算への入力が相互に対応する場合に、相互に対応する。その場合、出力が対応するための条件は、入力が対応するための条件の論理積となる。

【００３１】算術演算とは異なり、ブール演算は完全に解釈される。これにより、条件を超えて演算を移動させるような変換の正当性のチェックが可能となる。このような変換は、スケジューリングにおいて一般的である。

【００３２】スケジューリングを検証する作業は、ビヘイビア記述におけるループと、スケジューリング中のループ変換との存在によって、非常に複雑になる。本発明の重要な特徴は、ループと、スケジューリングにおけるループ変換とが存在する場合に、等価性チェッカによって、スケジュールとビヘイビアにおける信号の間の対応形式で、不変項の効率的な抽出を行うことである。本発明の技術は、ほとんどの設計における状態空間爆発は、制御状態よりもデータパスレジスタによって引き起こさ

れるという観測に、部分的に基づいている。スケジューリングにおける代表的な変換とみなされるものに基づいて、本発明の技術は、スケジューリングによって生成されるほとんどの設計を検証することが可能である。本発明の技術は、扱うことができないループ最適化に遭遇した場合に、誤った否定（フォールスネガティブ）を報告するという点で、悲観的である。本発明の技術の詳細については、その応用の具体例とともに、セクション４で説明する。

【００３３】シンボリックシミュレーションアルゴリズムは、本発明の重要な構成要素であるが、本発明の主要な貢献ではない。本発明の主要な貢献は、ループを扱うことが可能な、無閉路グラフに対する基本的なシンボリックシミュレーションアルゴリズムの改善にある。

【００３４】従来の方法における問題点を解決するために、本発明の目的は、スケジュールと、そのビヘイビア記述との等価性を証明する改善された方法を提供することである。本発明は、いかなるスケジュールにも制限されず、従来の技術の項で説明したいかなる最適化がなされたスケジュールでも使用可能である。なお、従来の技術の項で説明した最適化は単なる例示であり、本発明は、他の最適化技術を適用したスケジュールにも適用可能である。

【００３５】ビヘイビアは、従来の任意の形式で指定することができる。これには、制御フローグラフ、データフローグラフあるいは制御／データフローグラフ（ＣＤＦＧ：control/data flow graph）及びビヘイビア（超）状態マシンが含まれるが、これらには限定されない。ビヘイビア合成についての詳細は、D. Knapp, T. Ly, D. MacMillen, and R. Miller, "Behavioral synthesis methodology for HDL-based specification and validation", in Proc. Design Automation Conf., pp. 28-291, June 1995、を参照。

【００３６】本発明は、ビヘイビア及びスケジュールにおけるプライマリ入力変数の間の対応が与えられていること、及び、出力変数間の対応と、出力変数が同一の値を有すると期待される時刻とが明確に指定されていることを仮定する。本発明は、複数のループ、ネストしたループ、及びデータ依存ループを含むビヘイビア及びスケジュールを処理する。

【００３７】本発明の検証手続きの正確さ及び完全性を保証するために設計及び合成のフローが満たすことが必要とされる仮定は以下の通りである。

【００３８】・ビヘイビア記述における演算は、スケジューリングプロセス中にアトミックエンティティとして扱われるもの（例えば、算術及び比較演算）と、分解または変換される可能性のあるもの（例えば、ブール演算）とに分けることができる。例えば、ワードあるいはビットベクタ演算（例えば加算）は、スケジューリングプロセス中には、そのゲートレベル実装に分解されないことがある。アトミック演算と非アトミック演算に演算を分けることは任意性を伴うことがあるが、検証手続きに与えられることが必要である。この情報は、本発明の検証技術の主要な構成要素である非解釈シンボリックシミュレーション手続きにより、どの演算を解釈しどの演算を非解釈のまま残すべきかを決定するために使用される。

【００３９】・スケジューリングプロセスは、アトミック演算の解釈から導き出される知識を使用しない。例えば、算術及び比較演算がアトミックであると宣言される場合、スケジューリングは、スケジュールを最適化するために、これらの演算の機能についての知識を使用しない。比較演算は、分岐及びループ終了条件を決定するために使用されるものを含む。

【００４０】・ビヘイビアにおける各ループごとに、スケジュールには少なくとも１つの対応するループがあり、スケジュールにおけるループの１回の実行は、ビヘイビアにおけるループの１回以上の実行に対応する。この性質を満たさないスケジュールは、検証手続きによってエラーありとしてフラグが立てられる。なお、この仮定は、ループ本体あるいは境界がビヘイビアとスケジュールとで同一であることを要求するものではない。むしろ、これは、ループ展開がビヘイビアからスケジュールへと実行されているだけであり、その逆ではないことを意味する。

【００４１】上記の仮定はそれほど制限的ではない。その理由は、これらの仮定は、リストスケジューリング、強制的スケジューリング(force-directed scheduling)、パスに基づくスケジューリング、ループ指向スケジューリング(loop-directed scheduling)などのような周知のスケジューリングアルゴリズムを含む最も実際的なスケジューリング技術によって満たされるからである。これについては、

. D. D. Gajski, N. D. Dutt, A. C.-H. Wu, and S. Y.-L. Lin, High-levelSynthesis: Introduction to Chip and System Design, Kluwer Academic Publishers, Norwell, MA, 1992

・G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York, NY, 1994

に示されている。

【００４２】本明細書では、代表的なスケジューリング技術という用語は、上記の仮定を満たすスケジューリングのアルゴリズムあるいはツールを表すために使用する。

【００４３】ループについて正当性をチェックする場合、本発明のアプローチはループ不変項を使用するものである。しかし、ループ停止の問題、すなわち、ループ本体の後のコードが実際に実行されるかどうかには特に対処しない。ある意味で、本発明では、すべてのｎ≧０について、ｎ回の反復後の停止を考え、すべての場合に

等価性をチェックする。本発明のアプローチのこの特徴は強調しなければならない。すなわち、ループ本体のすべての反復回数に対する等価性がチェックされる。なお、算術演算を扱うために非解釈関数を使用するため、解釈された値（これがスケジューラによって利用されたかどうかにかかわらず）に依存する停止条件を考慮することはこのフレームワークでは不可能である。例えば、ループが６回実行される場合に限りエラーが生じるが、終了条件のために、ループは２回より多くは決して実行されないとする。この場合、本発明の手続きは、フォールスネガティブを報告することになる。その理由は、本発明は、２回の実行後の停止のみならず、ｎ＝６を含むすべての回数ｎの後の停止を考慮するからである。ループ反復回数が一定の上限を有するような場合を早期停止(early termination)という。

【００４４】

【課題を解決するための手段】本発明の目的を達成するため、回路のスケジューリングの正当性をチェックする方法が提供される。回路に対するスケジュールは、ビヘイビア記述から得られる。この方法は、ループが回路内にあるときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出するステップと、ループ不変項を抽出するためにシンボリックシミュレーションを実行するステップと、非巡回スレッドの等価性を証明するステップとを有する。

【００４５】好ましくは、ビヘイビア記述は、サイクル境界の導入によって変換される。

【００４６】好ましくは、ビヘイビア記述は、演算並べ替えによって変換される。

【００４７】好ましくは、ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換される。

【００４８】好ましくは、ビヘイビア記述は、演算の投機実行によって変換される。

【００４９】本発明のもう１つの特徴によれば、回路のビヘイビア記述に対して回路のスケジュールを検証する方法が提供される。この方法は、前記スケジュールから、ループを含む可能性のある実行のスケジュールスレッドを選択するステップと、前記ビヘイビア記述から対応するビヘイビアスレッドを識別するステップと、スケジュールスレッドとビヘイビアスレッドの無条件等価性を証明するステップと、実行のすべてのスレッドについて繰り返すステップとを有する。

【００５０】好ましくは、スケジュールは、スケジュール状態遷移グラフとして指定される。

【００５１】好ましくは、ビヘイビアは、ビヘイビア状態遷移グラフとして指定される。

【００５２】好ましくは、前記証明するステップは、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップとを有する。

【００５３】本発明のもう１つの特徴によれば、回路のビヘイビア記述に対して回路のスケジュールを検証する方法が提供される。この方法は、スケジュールをスケジュール状態遷移グラフとして指定するステップと、回路のビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するステップと、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを識別するステップと、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、実行のすべてのスレッドについて繰り返すステップとを有する。

【００５４】好ましくは、等価性チェックは、前記ビヘイビア状態遷移グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記ビヘイビア構造グラフ内のすべてのノードを含む順序セットａｒｒ１を作成するステップと、前記ビヘイビア構造グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記スケジュール構造グラフ内のすべてのノードを含む順序セットａｒｒ２を作成するステップと、ａｒｒ１をたどり、ビヘイビア構造グラフ内の基底変数を識別するステップと、ビヘイビア構造グラフ内の非基底変数を基底変数で表すステップと、スケジュール構造グラフ内の入力ノードに対する等価性リストを構成するステップと、ａｒｒ２をたどり、ａｒｒ２内の各ノードを処理して、スケジュール構造グラフの入力からスケジュール構造グラフの出力へ等価性リストを伝搬させるステップと、各等価性リスト内のエントリは対（ｕ，ｃ）であり、ｕはビヘイビア構造グラフ内の信号の識別子であり、ｃは等価性の条件を表す二分決定ダイヤグラムであるとして、ビヘイビア構造グラフ内の対応する出力ノードで等価性が確定したかどうか、及び、対応する条件ｃがａｒｒ２内のプライマリ出力ノードに対するトートロジーであるかどうかをチェックするステップと、ａｒｒ２内のすべての出力ノードについて繰り返すステップと、すべての出力ノードが等価であることがわかった場合に等価性を見つけたとするステップとを有するプロセスによって行われる。

【００５５】本発明のもう１つの特徴によれば、回路のスケジュールと該回路のビヘイビアとの間の等価性を検証する方法が提供される。前記スケジュール及び前記ビヘイビアは、実行の巡回スレッドを有する可能性がある。前記方法は、スケジュールをスケジュール状態遷移

グラフとして表現するステップと、ビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、前記スケジュール状態遷移グラフ内の強連結成分を識別するステップと、各強連結成分内の終了ノードを識別するステップと、前記スケジュール状態遷移グラフをつぶして、前記強連結成分を通らないサブパスを併合するステップと、以前に選択されていないパスを選択するステップと、選択されたパスに対する構造ＲＴＬ回路を取得するステップと、選択されたパスを列挙するのに必要なすべての状態遷移決定をカプセル化するパスシグナルを生成するための回路を構造ＲＴＬ回路に追加するステップと、パスシグナルを用いて、制約されたシンボリックシミュレーションを実行してビヘイビア状態遷移グラフ内の対応するパスを識別し、該パスに対する構造ＲＴＬ回路を取得するステップと、選択されたパスにおいて、以前に選択されていない強連結成分を選択するステップと、選択されたパス内の選択された強連結成分に対する不変項を、対応セットのリストとして抽出するステップと、対応セットのリストから１つの対応セットを選択するステップと、選択された対応セットが、前のシンボリックシミュレーションの強連結成分カットにおいて得られる変数対応より小さい場合に、シンボリックシミュレーションを再実行するステップと、対応セットのリスト内の各対応セットについて以上のステップを繰り返すステップと、出力等価性条件が、パス条件以外の条件付きであるかどうかをテストするステップと、前記出力等価性が条件付きである場合に非等価性を報告してこの方法を終了するステップと、選択されたパス内のすべての強連結成分について以上のステップを繰り返すステップと、終了点が高々３度現れるようにルートからシンクへのすべてのパスについて以上のステップを繰り返すステップとを有する。

【００５６】好ましくは、制約されないシンボリックシミュレーションが、ビヘイビア状態遷移グラフの始状態を許容パスリストに割り当てるステップと、許容パスリスト内で以前に訪れていない状態を選択するステップと、ビヘイビア構造ＲＴＬを生成するステップと、非解釈シンボリックシミュレーションを実行して、スケジュール構造ＲＴＬ及びビヘイビア構造ＲＴＬ内の対応する信号を識別するステップと、遷移条件とパスシグナルの論理積がゼロでない場合に、状態S_jの新しいコピーを許容パスに追加するステップと、S_iからS_jへの各出遷移ごとに前記追加するステップを繰り返すステップと、許容パス内に残る訪れていない状態のみが終状態のインスタンスとなるまで、すべての訪れていない状態について繰り返すステップとを有するプロセスを用いて実行される。

【００５７】好ましくは、不変項は、各ループごとに、各カットが前記ループの各実行の境界における変数値を表すような、スケジュール内のパスの構造ＲＴＬ回路内

の３個のカットを識別するステップと、ビヘイビアにおけるパスの構造ＲＴＬ回路内の対応するカットを識別して、第１と第２のカットの間のサブ回路と、第２と第３のカットの間のサブ回路が同型であることをチェックするステップと、スケジュール及びビヘイビアのＲＴＬ回路における対応するカットの各対における変数どうしの間の等価関係を識別するステップと、最後のカットと最後の前のカットとの間の等価関係が同一であるかどうかをチェックするステップと、前記関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットである場合、最後の前のカットにおける等価関係を破棄し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、繰り返すステップと、前記関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットでない場合、最後の前のカットにおける等価関係を、等価関係セットの集合に追加し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、繰り返すステップと、前記関係が同一である場合、最後のカットにおける等価関係を、等価関係セットの集合に追加するステップと、等価関係セットの集合内で、他のエントリのスーパーセットであるすべてのエントリを削除するステップと、等価関係セットの最終集合を、不変項の所望の集合として指定するステップとを有するプロセスを用いて、ループから抽出される。

【００５８】本発明のもう１つの特徴によれば、回路のスケジューリングの正当性をチェックするシステムが提供される。回路に対するスケジュールは、ビヘイビア記述から得られる。このシステムは、ループが存在するときに非巡回スレッドの十分なセットを決定するループ不変項抽出器と、前記ループ不変項を抽出するシンボリックシミュレータと、非巡回スレッドの等価性を証明する等価性証明器とを有する。

【００５９】好ましくは、前記ビヘイビア記述は、サイクル境界の導入によって変換される。

【００６０】好ましくは、前記ビヘイビア記述は、演算並べ替えによって変換される。

【００６１】好ましくは、前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換される。

【００６２】好ましくは、前記ビヘイビア記述は、演算の投機実行によって変換される。

【００６３】本発明のもう１つの特徴によれば、回路のビヘイビア記述に対して回路のスケジュールを検証するシステムが提供される。このシステムは、スケジュールをスケジュール状態遷移グラフとして指定するスケジュール状態遷移グラフジェネレータと、回路のビヘイビアをビヘイビア状態遷移グラフとして指定するビヘイビア状態遷移グラフジェネレータと、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択す

るスケジュールスレッドセレクタと、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを選択するビヘイビアスレッドセレクタと、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するコンバータと、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックする等価性チェッカとを有する。

【００６４】本発明のもう１つの特徴によれば、回路のスケジューリングの正当性をチェックするための、プロセッサ及びメモリを有するコンピュータシステムが提供される。回路に対するスケジュールは、ビヘイビア記述から得られる。前記メモリは、前記コンピュータシステムが前記チェックを実行することを可能にする命令を含み、該命令は、ループが存在するときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出する命令と、ループ不変項を抽出するためのシンボリックシミュレーションの命令と、非巡回スレッドの等価性を証明する命令とを含む。

【００６５】好ましくは、前記ビヘイビア記述は、サイクル境界の導入によって変換される。

【００６６】好ましくは、前記ビヘイビア記述は、演算並べ替えによって変換される。

【００６７】好ましくは、前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換される。

【００６８】好ましくは、前記ビヘイビア記述は、演算の投機実行によって変換される。

【００６９】本発明のもう１つの特徴によれば、回路のビヘイビア記述に対して回路のスケジュールを検証するための、プロセッサ及びメモリを有するコンピュータシステムが提供される。前記メモリは、前記コンピュータシステムが前記検証を実行することを可能にする命令を含み、該命令は、スケジュールをスケジュール状態遷移グラフとして指定する命令と、回路のビヘイビアをビヘイビア状態遷移グラフとして表現する命令と、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択する命令と、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを選択する命令と、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換する命令と、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックする命令と、実行のすべてのスレッドについて繰り返す命令とを含む。

【００７０】本発明のもう１つの特徴によれば、回路のビヘイビア記述に対して回路のスケジュールを検証するための、プロセッサ及びメモリを有するコンピュータシステムが提供される。前記メモリは、前記コンピュータシステムが、スケジュールをスケジュール状態遷移グラフとして指定するステップと、回路のビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するステップと、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを識別するステップと、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、実行のすべてのスレッドについて繰り返すステップとを実行することを可能にする命令を含む。

【００７１】好ましくは、前記命令は、前記コンピュータシステムが、前記ビヘイビア状態遷移グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記ビヘイビア構造グラフ内のすべてのノードを含む順序セットａｒｒ１を作成するステップと、前記ビヘイビア構造グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記スケジュール構造グラフ内のすべてのノードを含む順序セットａｒｒ２を作成するステップと、ａｒｒ１をたどり、ビヘイビア構造グラフ内の基底変数を識別するステップと、ビヘイビア構造グラフ内の非基底変数を基底変数で表すステップと、スケジュール構造グラフ内の入力ノードに対する等価性リストを構成するステップと、ａｒｒ２をたどり、ａｒｒ２内の各ノードを処理して、スケジュール構造グラフの入力からスケジュール構造グラフの出力へ等価性リストを伝搬させるステップと、各等価性リスト内のエントリは対（ｕ，ｃ）であり、ｕはビヘイビア構造グラフ内の信号の識別子であり、ｃは等価性の条件を表す二分決定ダイヤグラムであるとして、ビヘイビア構造グラフ内の対応する出力ノードで等価性が確定したかどうか、及び、対応する条件ｃがａｒｒ２内のプライマリ出力ノードに対するトートロジーであるかどうかをチェックするステップと、ａｒｒ２内のすべての出力ノードについて繰り返すステップと、すべての出力ノードが等価であることがわかった場合に等価性を見つけたとするステップとを実行することを可能にする命令をさらに含む。

【００７２】本発明のもう１つの特徴によれば、回路のスケジュールと該回路のビヘイビアとの間の等価性を検証するための、プロセッサ及びメモリを有するコンピュータシステムが提供される。前記スケジュール及び前記ビヘイビアは、実行の巡回スレッドを有する可能性がある。前記メモリは、前記コンピュータシステムが、スケジュールをスケジュール状態遷移グラフとして表現するステップと、ビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、前記スケジュール状態遷移グラフ内の強連結成分を識別するステップと、各強連結成分内の終了ノードを識別するステップと、前記スケジュ

ール状態遷移グラフをつぶして、前記強連結成分を通らないサブパスを併合するステップと、以前に選択されていないパスを選択するステップと、選択されたパスに対する構造ＲＴＬ回路を取得するステップと、選択されたパスを列挙するのに必要なすべての状態遷移決定をカプセル化するパスシグナルを生成するための回路を構造ＲＴＬ回路に追加するステップと、パスシグナルを用いて、制約されたシンボリックシミュレーションを実行してビヘイビア状態遷移グラフ内の対応するパスを識別するステップと、選択されたパスにおいて、以前に選択されていない強連結成分を選択するステップと、選択されたパス内の選択された強連結成分に対する不変項を、対応セットのリストとして抽出するステップと、対応セットのリストから１つの対応セットを選択するステップと、選択された対応セットが、前のシンボリックシミュレーションの強連結成分カットにおいて得られる変数対応より小さい場合に、シンボリックシミュレーションを再実行するステップと、対応セットのリスト内の各対応セットについて以上のステップを繰り返すステップと、出力等価性条件が、非等価性を報告するパス条件以外の条件付きであるかどうかをテストし、前記出力等価性が条件付きである場合にこの検証を終了するステップと、選択されたパス内のすべての強連結成分について以上のステップを繰り返すステップと、終了点が高々３度現れるようにルートからシンクへのすべてのパスについて以上のステップを繰り返すステップとを用いて前記検証を実行することを可能にする。

【００７３】好ましくは、前記命令は、前記コンピュータシステムが、ビヘイビア状態遷移グラフの始状態を許容パスリストに割り当てるステップと、許容パスリスト内で以前に訪れていない状態を選択するステップと、ビヘイビア構造ＲＴＬを生成するステップと、非解釈シンボリックシミュレーションを実行して、スケジュール構造ＲＴＬ及びビヘイビア構造ＲＴＬ内の対応する信号を識別するステップと、遷移条件とパスシグナルの論理積がゼロでない場合に、状態S_jの新しいコピーを許容パスに追加するステップと、S_iからS_jへの各出遷移ごとに前記追加するステップを繰り返すステップと、許容パス内に残る訪れていない状態のみが終状態のインスタンスとなるまで、すべての訪れていない状態について繰り返すステップとを実行することを可能にする命令をさらに含む。

【００７４】好ましくは、前記命令は、前記コンピュータシステムが、各ループごとに、各カットが前記ループの各実行の境界における変数値を表すような、スケジュール内のパスの構造ＲＴＬ回路内の３個のカットを識別するステップと、ビヘイビアにおけるパスの構造ＲＴＬ回路内の対応するカットを識別して、第１と第２のカットの間のサブ回路と、第２と第３のカットの間のサブ回路が同型であることをチェックするステップと、スケジュール及びビヘイビアのＲＴＬ回路における対応するカットの各対における変数どうしの間の等価関係を識別するステップと、最後のカットと最後の前のカットとの間の等価関係が同一であるかどうかをチェックするステップと、前記関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットである場合、最後の前のカットにおける等価関係を破棄し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、繰り返すステップと、前記関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットでない場合、最後の前のカットにおける等価関係を、等価関係セットの集合に追加し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、繰り返すステップと、前記関係が同一である場合、最後のカットにおける等価関係を、等価関係セットの集合に追加するステップと、等価関係セットの集合内で、他のエントリのスーパーセットであるすべてのエントリを削除するステップと、等価関係セットの最終集合を、不変項の所望の集合として指定するステップとを実行することを可能にする命令をさらに含む。

【００７５】本発明のもう１つの特徴によれば、コンピュータが回路のスケジューリングの正当性をチェックすることを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品が提供される。回路に対するスケジュールは、ビヘイビア記述から得られる。前記コンピュータコードは、ループが存在するときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出するコンピュータコードと、ループ不変項を抽出するためのシンボリックシミュレーションのコンピュータコードと、非巡回スレッドの等価性を証明するコンピュータコードとを含む。

【００７６】好ましくは、前記ビヘイビア記述は、サイクル境界の導入によって変換される。

【００７７】好ましくは、前記ビヘイビア記述は、演算並べ替えによって変換される。

【００７８】好ましくは、前記ビヘイビア記述は、ループの展開、巻付け、折畳み及びパイプライン化によって変換される。

【００７９】好ましくは、前記ビヘイビア記述は、演算の投機実行によって変換される。

【００８０】本発明のもう１つの特徴によれば、コンピュータが回路のビヘイビア記述に対して回路のスケジュールを検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品が提供される。前記コンピュータコードは、前記コンピュータが、スケジュールをスケジュール状態遷移グラフとして指定することを可能にするスケジュール状態遷移グラフジェネレータコードと、前記コンピュータが、回路のビヘイビアをビヘイビア状態遷移グラフと

して指定することを可能にするビヘイビア状態遷移グラフジェネレータコードと、前記コンピュータが、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択することを可能にするスケジュールスレッドセレクタコードと、前記コンピュータが、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを選択することを可能にするビヘイビアスレッドセレクタコードと、前記コンピュータが、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換することを可能にするコンバータコードと、前記コンピュータが、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックすることを可能にする等価性チェッカコードとを含む。

【００８１】本発明のもう１つの特徴によれば、コンピュータが回路のビヘイビア記述に対して回路のスケジュールを検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品が提供される。前記コンピュータコードは、前記コンピュータが、スケジュールをスケジュール状態遷移グラフとして指定するステップと、回路のビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、前記スケジュール状態遷移グラフから、実行のスケジュールスレッドを選択するステップと、前記ビヘイビア状態遷移グラフから、対応するビヘイビアスレッドを識別するステップと、前記スケジュールスレッドをスケジュール構造グラフに変換するとともに前記ビヘイビアスレッドをビヘイビア構造グラフに変換するステップと、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックするステップと、実行のすべてのスレッドについて繰り返すステップとを実行することを可能にする。

【００８２】好ましくは、前記コンピュータコードは、前記コンピュータが、前記ビヘイビア状態遷移グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記ビヘイビア構造グラフ内のすべてのノードを含む順序セットａｒｒ１を作成するステップと、前記ビヘイビア構造グラフ内の各ノードが該ノードの推移ファンイン内のすべてのノードの後にのみ現れるように、前記スケジュール構造グラフ内のすべてのノードを含む順序セットａｒｒ２を作成するステップと、ａｒｒ１をたどり、ビヘイビア構造グラフ内の基底変数を識別するステップと、ビヘイビア構造グラフ内の非基底変数を基底変数で表すステップと、スケジュール構造グラフ内の入力ノードに対する等価性リストを構成するステップと、ａｒｒ２をたどり、ａｒｒ２内の各ノードを処理して、スケジュール構造グラフの入力からスケジュール構造グラフの出力へ等価性リストを伝搬させるステップと、各等価性リスト内のエントリは対（ｕ，ｃ）であり、ｕはビヘイビア構造グラフ内の信号の識別子であり、ｃは等価性の条件を表す二分決定ダイヤグラムであるとして、ビヘイビア構造グラフ内の対応する出力ノードで等価性が確定したかどうか、及び、対応する条件ｃがａｒｒ２内のプライマリ出力ノードに対するトートロジーであるかどうかをチェックするステップと、ａｒｒ２内のすべての出力ノードについて繰り返すステップと、すべての出力ノードが等価であることがわかった場合に等価性を見つけたとするステップとを実行することを可能にする。

【００８３】本発明のもう１つの特徴によれば、コンピュータが回路のスケジュールと該回路のビヘイビアとの間の等価性を検証することを可能にするコンピュータコードを含むコンピュータ可読媒体を有するコンピュータプログラム製品が提供される。前記スケジュール及び前記ビヘイビアは、実行の巡回スレッドを有する可能性がある。前記コンピュータコードは、前記コンピュータが、スケジュールをスケジュール状態遷移グラフとして表現するステップと、ビヘイビアをビヘイビア状態遷移グラフとして表現するステップと、前記スケジュール状態遷移グラフ内の強連結成分を識別するステップと、各強連結成分内の終了ノードを識別するステップと、前記スケジュール状態遷移グラフをつぶして、前記強連結成分を通らないサブパスを併合するステップと、以前に選択されていないパスを選択するステップと、選択されたパスに対する構造ＲＴＬ回路を取得するステップと、選択されたパスを列挙するのに必要なすべての状態遷移決定をカプセル化するパスシグナルを生成するための回路を構造ＲＴＬ回路に追加するステップと、パスシグナルを用いて、制約されたシンボリックシミュレーションを実行してビヘイビア状態遷移グラフ内の対応するパスを識別し、該パスに対する構造ＲＴＬ回路を取得するステップと、選択されたパスにおいて、以前に選択されていない強連結成分を選択するステップと、選択されたパス内の選択された強連結成分に対する不変項を、対応セットのリストとして抽出するステップと、対応セットのリストから１つの対応セットを選択するステップと、選択された対応セットが、前のシンボリックシミュレーションの強連結成分カットにおいて得られる変数対応より小さい場合に、シンボリックシミュレーションを再実行するステップと、対応セットのリスト内の各対応セットについて以上のステップを繰り返すステップと、出力等価性条件が、パス条件以外の条件付きであるかどうかをテストするステップと、前記出力等価性が条件付きである場合に非等価性を報告してこの方法を終了するステップと、選択されたパス内のすべての強連結成分について以上のステップを繰り返すステップと、終了点が高々３度現れるようにルートからシンクへのすべてのパスについて以上のステップを繰り返すステップとを実行することを可能にする。

【００８４】好ましくは、前記コンピュータコードは、

前記コンピュータが、ビヘイビア状態遷移グラフの始状態を許容パスリストに割り当てるステップと、許容パスリスト内で以前に訪れていない状態を選択するステップと、ビヘイビア構造ＲＴＬを生成するステップと、非解釈シンボリックシミュレーションを実行して、スケジュール構造ＲＴＬ及びビヘイビア構造ＲＴＬ内の対応する信号を識別するステップと、遷移条件とパスシグナルの論理積がゼロでない場合に、状態S_jの新しいコピーを許容パスに追加するステップと、S_iからS_jへの各出遷移ごとに前記追加するステップを繰り返すステップと、許容パス内に残る訪れていない状態のみが終状態のインスタンスとなるまで、すべての訪れていない状態について繰り返すステップとを用いて、制約されないシンボリックシミュレーションを実行することを可能にする。

【００８５】好ましくは、前記コンピュータコードは、前記コンピュータが、各ループごとに、各カットが前記ループの各実行の境界における変数値を表すような、スケジュール内のパスの構造ＲＴＬ回路内の３個のカットを識別するステップと、ビヘイビアにおけるパスの構造ＲＴＬ回路内の対応するカットを識別して、第１と第２のカットの間のサブ回路と、第２と第３のカットの間のサブ回路が同型であることをチェックするステップと、スケジュール及びビヘイビアのＲＴＬ回路における対応するカットの各対における変数どうしの間の等価関係を識別するステップと、最後のカットと最後の前のカットとの間の等価関係が同一であるかどうかをチェックするステップと、前記関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットである場合、最後の前のカットにおける等価関係を破棄し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、繰り返すステップと、前記関係が同一でなく、かつ、最後のカットにおける等価関係が、最後の前のカットにおける等価関係のサブセットでない場合、最後の前のカットにおける等価関係を、等価関係セットの集合に追加し、１つ以上のループ実行について２つのＲＴＬ回路を展開して、繰り返すステップと、前記関係が同一である場合、最後のカットにおける等価関係を、等価関係セットの集合に追加するステップと、等価関係セットの集合内で、他のエントリのスーパーセットであるすべてのエントリを削除するステップと、等価関係セットの最終集合を、不変項の所望の集合として指定するステップとを用いて不変項を抽出することを可能にする。

【００８６】

【発明の実施の形態】［４．好ましい実施形態］

［４．１　非巡回の場合の非解釈関数によるシンボリックシミュレーション］このサブセクション（４．１）では、有限長の非巡回スレッドを有するスケジュールの場合に、ビヘイビア記述に対してスケジュールを検証する際の、非解釈シンボリックシミュレーションアルゴリズムの好ましい実施例について説明する。このサブセクションで提示されるすべての説明では、スケジュール及びビヘイビアにおける有限長の非巡回スレッドのみを比較する。

【００８７】次のサブセクション（４．２）では、ループを含むスレッドを比較するという一般的な場合を扱う検証手続きの好ましい実施例について説明する。

【００８８】４．１．１　スケジュールの表現

この好ましい実施例では、スケジュールは、スケジュール状態遷移グラフ（スケジュールＳＴＧ：Schedule State Transition Graph）の形で指定される。スケジュールＳＴＧは、拡張有限状態マシン（ＥＦＳＭ：Extended Finite StateMachine）やビヘイビア有限状態マシン（ＢＦＳＭ：Behavioral Finite State Machine）表現によく似ている。ＥＦＳＭについて詳細には、K. T. Cheng and A.S. Krishnakumar, "Automatic functional test generation using the extended finite state machine model", in Proc. Design Automation Conf., June 1993、を参照。ＢＦＳＭ表現について詳細には、O. Lakshminarayana, A. Raghunathan, and N. K. Jha, "Incorporating speculative execution into scheduling for control-flow intensive behaviors", in Proc. Design Automation Conf., pp.108-113, June 1998、を参照。ＦＳＭ表現について詳細には、W. Wolf, A. Takach, C. Huang, and R. Mano, "The Princeton University behavioral synthesis system", in Proc. Design Automation Conf., pp.182-187, June 1992、を参照。スケジュールＳＴＧは、機能ＲＴＬコードが各状態に埋め込まれた、状態遷移グラフからなる。

【００８９】ＳＴＧの状態S_i内に埋め込まれたコードは、S_iで実行されることが必要なデータパス演算を指定するとともに、S_iからのそれぞれの出状態遷移に対する遷移条件を計算する。状態遷移グラフの状態内に埋め込まれたコードは、変数Ｖ、演算Ｏ、及びクロックにより定義することができる。クロックは、変数の値の更新を支配する。

【００９０】変数は、Ｖ＝（ＰＩ，ＰＯ，Ｒ，Ｔ）のように４つのセットに分けられる。ＰＩはプライマリ入力のセットであり、ＰＯはプライマリ出力のセットであり、Ｒはレジスタ変数のセットであり、Ｔは一時変数のセットである。演算は、制御演算のセット、及び、代入演算のセットを含む。制御演算の結果はブール型であり、この結果は、他の演算の実行を制御するために使用される。代入演算は、変数に値を代入し、あるいは、変数の値を変更する。各演算op_iは、それに関連づけられた対応する条件c_iを有する。条件c_iは論理式であり、他の代入及び制御演算の結果を用いて構成されることが可能である。演算op_iは、c_iが真の場合に限り実行される。状態遷移は、状態内に埋め込まれたコード内のgoto文によって指示されることも可能である。それぞ

れのこのようなgoto文の実行条件は、対応する状態遷移条件を表す。

【００９１】スケジュールは、明確に定義された始状態及び終状態を有すると仮定する。スケジュールの実行は、始状態で開始し、終状態で終了する。なお、複数の可能な終状態がある場合は、スケジュールのすべての終状態から入ってくる弧を有するダミー状態をスケジュールに追加し、このダミー状態のみを終状態と見なすことによって、単純な場合に帰着される。

【００９２】４．１．２　ビヘイビアの表現

さまざまなハイレベル合成ツールが、制御フローグラフ（ＣＦＧ：control flow graph）、データフローグラフ（ＤＦＧ：data flow graph）、制御／データフローグラフ（ＣＤＦＧ）、及びビヘイビア有限状態マシン（ＢＦＳＭ）を含むビヘイビア記述に対するさまざまな表現を使用している。本発明は、スケジュール表現の特定の方法に制限されない。これら及びその他のうちのいずれの表現も、検証手続きのために使用可能である。説明の簡単化及び一貫性のために、好ましい実施例は、ビヘイビア及びスケジュールを表すために同じデータ構造を使用する。

【００９３】このため、好ましい実施例では、ビヘイビアはビヘイビア状態遷移グラフ（ビヘイビアＳＴＧ）として表現される。ビヘイビアＳＴＧは、ビヘイビアから直接に導出されることを除いては、上記のスケジュールＳＴＧと類似している。ループを含まないビヘイビアの部分は、単一の状態にまとめられる。従って、非巡回ビヘイビアは、ただ１つの状態を有するビヘイビアＳＴＧに翻訳することができる。このようにビヘイビア及びスケジュールに対する一貫したデータ構造を有することにより、等価性チェックの問題は、スケジュールＳＴＧとビヘイビアＳＴＧの等価性を証明することに帰着する。

【００９４】・ビヘイビアＳＴＧ（ＢＳＴＧ）及びスケジュールＳＴＧ（ＳＳＴＧ）。

・ＢＳＴＧとＳＳＴＧにおけるプライマリ入力変数どうしの間の対応。

が与えられた場合、目標は、対応する出力変数においてＢＳＴＧ及びＳＳＴＧによって生成される値が等しいことを証明することである。

【００９５】前述のように、この好ましい実施例では、焦点は、ＢＳＴＧ及びＳＳＴＧにおける実行の個々のスレッドどうしの間の等価性を証明するという制限された問題にある。ＳＳＴＧ（あるいはＢＳＴＧ）における実行のスレッドとは、始状態に始まり終状態に終わる状態遷移グラフにおける有限長のパスのことである。なお、パスが単純であることは要求されない。すなわち、パスは、状態遷移グラフ内のサイクルを一定有限回通ることも可能である。

【００９６】ＳＳＴＧ（あるいはＢＳＴＧ）における実行のスレッドが与えられると、その中で実行される計算は、構造グラフに変換される。

【００９７】定義１（構造グラフ）：構造グラフとは、有向グラフＧ＝（Ｖ，Ａ）であって、頂点のセットＶは演算を実行するハードウェアコンポーネントを表し、辺のセットはコンポーネントの構造連結性を表すものである。頂点∈Ｖは、型属性を有し、これは以下の値をとりうる。

・ＩＮ（プライマリ入力変数と、レジスタ変数の現サイクル値とを表す）

・ＯＵＴ（プライマリ出力変数と、レジスタ変数の次サイクル値とを表す）

・ＯＰ（算術演算及び比較演算を含む、アトミックなワードレベル演算を表す）

・ＬＯＧＩＣ（制御またはランダム論理を表す）

・ＭＵＸ

構造グラフ内の辺にはそのビット幅が標記(annotate)される。

【００９８】計算のセットから構造グラフを構成するプロセスは、ハードウェア記述言語（ＨＤＬ：Hardware Description Language）からハードウェア構造を推論することと類似している。ＩＮ及びＯＵＴノードは、プライマリ入出力変数、定数値、及び、レジスタ変数の現サイクル及び次サイクルの値を表すように生成される。ＯＰノードは、ワードレベル計算及び条件演算（例えば、比較演算、case演算など）に関連する代入演算に対応して生成される。単一ビットまたはビットベクタに対するブール演算の使用により、構造グラフ内のＬＯＧＩＣノードが生成される。ＭＵＸノードは、相異なる代入文が、相異なる条件下で同じ変数に代入を行うときに構成される。これらの条件に対応するＯＰまたはＬＯＧＩＣノードの出力は、与えられたクロックサイクルにおいて実行される代入を決定するために、ＭＵＸノードへの選択（セレクト）入力として使用される。

【００９９】ＳＳＴＧ内の実行のスレッドＴと、等価であることを証明することが要求されるＢＳＴＧ内の対応する実行のスレッドＴ′とが与えられると、各スレッドに沿って実行される計算はまず構造グラフに変換される。こうして、問題は、２つの構造グラフSSG_Tと$BSG_{T'}$の等価性を証明することに帰着する。

【０１００】このセクションの残りの部分では、以下の性質を利用した、構造グラフの等価性チェックのためのアルゴリズムの好ましい実施例について説明する。

・ビヘイビア記述からスケジュールを生成するときにＯＰノードのアトミック性は保存される。

・算術変換（例えば、分配則や、乗算をシフトと加算で置き換えることなど）は実行されないということ。

【０１０１】定義２（条件付き等価性）：SSG_T内の信号ｖが$BSG_{T'}$内の信号$u_1, u_2, \ldots, u_n$に条件付き等価であるとは、対応する条件$c_1, c_2, \ldots, c_n$（条件とは、$BSG_{T'}$あるいはSSG_T内の入力変数へ

の値代入の空でないセットをを表す）であって、条件ｃkの下で、ＳＳＧT内の信号ｖにおける値が、ＢＳＧT' 内の信号ｕkにおける値と等しいことが保証されるような条件ｃ1，ｃ2，...，ｃnが存在する場合をいう。条件付き等価関係を表すために、記法

【数１】

$$V \cong \{(u_1, c_1), \ldots, (u_n, c_n)\}$$

を用いる。

【０１０２】ＢＤＤは、条件付き等価関係に関連する条件を表すために使用される。一般に、条件自体は、入力変数で表すことも可能であり、また、さまざまな算術及び条件演算の結果を含むことも可能である。しかし、条件は、ＩＮノードに加えて、ＯＰ及びＭＵＸノードの出力で（これらをまとめて、「基底変数で」という）表現される。実際、ＢＤＤは、制御論理に対してのみ構成される。これは、ＰＯに送られる次状態論理$R_{state\text{-}next}$と、ＭＵＸノードを通るどのパスがセンシタイズされているかあるいはマルチファンクションＦＵがどのように設定されているかを決定する論理とを含む。

【０１０３】ＢＳＧT' とＳＳＧTを比較するアルゴリズムの好ましい実施例の擬似コードを図４に示す。アルゴリズムは、ＳＳＧTとＢＳＧT' のＩＮノードどうしの間の等価関係からはじまる。このアルゴリズムは、ＰＯノードに到達するまでＳＳＧT内の中間信号を通って条件付き等価関係を生成し伝搬させ、ＳＳＧT及びＢＳＧT' における出力信号どうしの間の無条件等価性をチェックする。

【０１０４】まず、順序セットArr1（Arr2）を、ＢＳＧT' （ＳＳＧT）内のすべてのノードを含むように構成する。後方深さ優先探索走査を用いて、各ノードは、その推移ファンイン(transitive fanin)内のすべてのノードの後にのみ現れるようにされる。次に、ＢＳＧT' 内の基底変数を、ＰＩ、ＯＰ、及びＭＵＸノードの出力として識別する。次に、Arr1を通る走査を実行し、基底変数に対応しない出力を有する各ノード（すなわち、各ＬＯＧＩＣノード）について、そのノードの出力に対するＢＤＤを、その入力におけるＢＤＤに関して取得する。各ＳＳＧTノードは、その出力と、ＢＳＧT' 内の信号との間の条件付き等価関係を表す等価性リストに関連づけられる。等価性リスト内のエントリは対（ｕ，ｃ）である。ただし、ｕはＢＳＧT' 信号の識別子であり、ｃは、等価性のための条件を表すＢＤＤである。ＢＳＧT' とＳＳＧTの入力どうしの間の対応を用いて、ＳＳＧT内のＩＮノードに対する等価性リストを生成する。次に、Arr2を走査し、各ノードを、その入力から出力へ等価性リストを伝搬させるように処理する。ＯＰ、ＬＯＧＩＣ、及びＭＵＸノードを通って等価性リストを伝搬させる技術については後述する。ＳＳＧTのＰＯノードに到達すると、

アルゴリズムは、ＢＳＧT' 内の対応するＯＵＴノードで等価性が確定しているか、及び、対応する条件がトートロジーであるかどうかをチェックする。そうでない場合、アルゴリズムは、ＳＳＧTとＢＳＧT' は等価でないと報告する。ＳＳＧTのすべてのＯＵＴノードに対して無条件等価性が得られた場合に限り、アルゴリズムは、ＳＳＧTとＢＳＧT' が等価であると宣言する。

【０１０５】等価関係は、ＯＰノードを通って以下のように伝搬する。ＳＳＧT内のＯＰノードｖと、同じ演算を実行するＢＳＧT' 内のＯＰノードｕで、ｖの入力がｕの対応する入力と条件付き等価関係を有するようなものが存在する。このような場合、ｖとｕの出力は、対応する入力の等価条件の論理積と等価である。ＬＯＧＩＣノードに遭遇した場合、等価性リストをその出力に伝搬させるのではなく、ＢＳＧT' 内の基底変数の関数としてその出力を表すようにＢＤＤを構成する。これを行う理由は、ＬＯＧＩＣノードはスケジュールにおいて変換または導入されることがあるため、ＳＳＧTとＢＳＧT' の等価性を証明するためには解釈される必要があるからである。２入力ＭＵＸノードの１（０）データ入力からその出力へ等価性リストを伝搬させることは、選択（セレクト）信号に対するＢＤＤを取得し、それ（その補数）と、データ入力の等価性リスト内のすべての条件との論理積をとることによって、行われる。

【０１０６】［４．２　一般的な場合のスケジュール検証アルゴリズム］このサブセクションでは、一般的な場合のアルゴリズムの好ましい実施例について説明する。このアルゴリズムのタスクは、スケジュール及びビヘイビアの出力間の無条件等価性を確定することである。ＳＴＧが非巡回（無閉路）であれば、セクション４．１のシンボリックシミュレーションに基づく等価性チェッカで十分である。フィードバック（ループ）が存在する場合、等価性チェックアルゴリズムが有用であるためには、ループが完了するまで反復せずに２つの記述の等価性を検証することが必要である。ループを扱うため、アルゴリズムは、ループ不変項を抽出する。不変項は、ループ終了点におけるスケジュールとビヘイビアの間の変数対応である。不変項抽出は、等価性の証明を生成するためにループを完了まで反復することを不要にする自動帰納法に基づく。すべてのループ不変項が抽出されない場合、等価性チェッカはフォールスネガティブを返す可能性が高い。等価性チェックアルゴリズムは、スケジューリングが前に定義した意味で代表的である場合、すべてのループ不変項を検出し、真の否定及び肯定を返すことを保証する。このアルゴリズムは、誤った肯定を返すことがないという意味で安全である。本発明の検証アルゴリズムについて説明するための例を提示し、その後でその詳細について説明する。

【０１０７】４．２．１　具体例

アルゴリズムは、入力として、ビヘイビア及びスケジュ

ールの状態遷移グラフ（ＳＴＧ）表現（それぞれＢＳＴＧ及びＳＳＴＧ）をとる。ＳＴＧに加えて、プライマリ入力及び出力の対応のリストも、アルゴリズムに入力として提供される。アルゴリズムは、ＳＳＴＧ内の小さいパスセットを列挙することによって動作する。これらのパスは、ＳＳＴＧとＢＳＴＧの間の等価性を証明するための基礎として使用される。

【０１０８】例３：図５（ａ）に、簡約(reduced)ＳＳＴＧ（強連結成分を抽出し無閉路パスをつぶしたもの）の例を示す。このＳＳＴＧに対して、次の状態列を列挙することができる。

｛ＡＥ，ＡＢＣＥ，ＡＢＣＤＣＥ，ＡＢＣＤＣＤＣＥ，...｝

なお、パス｛ＡＢＣＤＣＥ，ＡＢＣＤＣＤＣＥ，...｝は、ループ本体の異なる回数の実行に対応する。これらのパスのすべてをＢＳＴＧ上でシミュレートする必要はない。ノード｛Ｃ｝は、ループ終了点に対応する。

｛Ｃ｝が０、１、及び３回現れるパスの数を列挙する。これらは、ループに全く遭遇しないこと、ループ終了条件に遭遇するがループ本体には遭遇しないこと、及び、ループ本体を２回実行すること、にそれぞれ対応する。最初の２つは単純パスであり、明確に列挙すべきである。終了パスが３回現れるパスを列挙する理由は、ループ本体を２回実行することによってループ不変項の生成に対する問題を設定することである。従って、この例で列挙されるパスは｛ＡＥ，ＡＢＣＥ，ＡＢＣＤＣＤＣＥ｝である。

【０１０９】これらのパスのそれぞれについて、ＢＳＴＧ内の対応するパスをシンボリックシミュレーションにより取得する。次に、アルゴリズムは、ＳＳＴＧ及びＢＳＴＧの対応するパスが等価であることを証明する。ループ本体を含むパス｛ＡＢＣＤＣＤＣＥ｝に対して、アルゴリズムはさらに進み、ループ本体内の演算が任意回数実行された場合に、ＳＳＴＧ及びＢＳＴＧの対応するパスどうしの間の等価関係が依然として維持されるかどうかを帰納的に証明する。これを行うため、アルゴリズムは、カット点｛ＡＢＣＤＣ｝及び｛ＡＢＣＤＣＤＣ｝での変数対応を抽出する。この場合、カット点｛ＡＢＣＤＣ｝及び｛ＡＢＣＤＣＤＣ｝における対応のセットは同一のままである。従って、帰納法により、列｛ＤＣ｝を任意個数だけ｛ＡＢＣＤＣＤＣ｝に連接しても依然として変数対応は維持されるということができる。従って、｛ＡＢＣＤＣＤＣＥ｝と、対応するＢＳＴＧパスが等価である場合、ループ本体の任意回の反復に対しても、ＳＳＴＧとＢＳＴＧは依然として等価になる。

【０１１０】次に、ループ終了に対応するカット点での対応する変数のセットが同一のままにならないような、別のシナリオを考える。セットが変わると、誤った結果を避けるために、収束するまで反復する必要がある。

【０１１１】例４：図６に示す例は、フォールスポジティブ（誤った肯定）を避けるために収束するまで反復する必要がある理由を例示する。図６の（ａ）及び（ｂ）は、それぞれ、回路のビヘイビア及びスケジュールを示す。なお、これらの２つは、スケジュールの状態５における文ｃ＝ｄ＋２のため、対応しない。最初に識別されるループ本体を含むパスは｛１，２，３，４，５，２，３，４，５，２，６｝である。図６（ｂ）における状態２は、ループ終了として識別され、状態３、４及び５はループ本体として識別される。ループの１回の実行のシンボリックシミュレーションの後、得られる変数対応は｛ａ≡ｐ，ｂ≡ｑ，ｄ≡ｓ｝であり、２回の実行の後には｛ｂ≡ｑ｝である。なお、ループの後の状態６の文（out = b）をシミュレートするための対応セットとしてこれらのいずれを用いても、ｂとｑが等価であると見なされていることによりフォールスポジティブを引き起こすことになる。

【０１１２】３回の実行の後にはじめて、手続きはｂとｑが対応しないと判定し、スケジュールとビヘイビアは非等価であると見なすことができる。このように、この場合、収束に到達するには、最初のパスのもう１回の反復をシミュレートしなければならない。すなわち、パス｛１，２，３，４，５，２，３，４，５，２，３，４，５，２，６｝もシミュレートする必要がある。

【０１１３】例５：最後にもう１つのシナリオを考える必要がある。前の例で、ループ反復後の対応のセットは、収束に到達するまで単調に減少した。しかし、一般に、反復を多く実行するにつれてこのセットが任意に変化するような例が考えられる。なお、実質的な目標は、ループ本体のすべての反復回数について等価性をチェックすることである。従って、収束が得られるまですべての極小対応セットを追跡しなければならない。これらは、他の対応セットのスーパーセットでないセットに対応する。ループ本体に続くコードは、収束後に得られる対応セットに対してシミュレートされるのに加えて、すべてのこのような極小セットに対してシミュレートされる。なお、この追加の対応によりシミュレートしないことによってフォールスポジティブを生じる可能性がある。後述するように、これらの極小対応セットを用いたシンボリックシミュレーションは、等価性をチェックするために必要十分である。

【０１１４】４．２．２　アルゴリズムの詳細

図７は、一般的な場合を扱う本発明の方法の好ましい実施例の擬似コードを示す。このアルゴリズムの第１のタスクは、ＳＳＴＧの、ループを構成する部分を識別することである。ループ不変項をループ終了点で計算する必要があるからである。ループは、強連結成分（ＳＣＣ：strongly connected component）を識別することによって見出される。各ＳＣＣは、１個以上の終了ノードを有し、そこからＳＣＣの外へ遷移することが可能である。その後、ＳＣＣを通らないサブパスを併合して、以後列

挙する必要のあるノード及びパスの総数を減らすためにＳＳＴＧをつぶす。図５（ａ）に、これらのステップによってＳＳＴＧがどのように影響されるかが示されている。これらのステップの結果、状態Ｃは、状態Ｃ及びＤからなるＳＣＣの終了点として識別される。

【０１１５】図７における擬似コードの第４行は、簡約ＳＳＴＧ内のパスを列挙するループの開始をマークする。このパスは、ＢＳＴＧ内の対応するパスに対してチェックされなければならない。パス列挙前にＳＳＴＧを簡約することにより、大幅にパスは少なくなる。図５（ａ）のＳＳＴＧの場合、最初に列挙される３個のパスは、次の状態列からなる。

｛ＡＥ，ＡＢＣＥ，ＡＢＣＤＣＤＣＥ｝

これらのパスのそれぞれについて、ＢＳＴＧ内の対応するパスをシンボリックシミュレーションにより取得する。図７の疑似コードの第５行は、列挙されたＳＳＴＧパスに対するＲＴＬ回路（ＳＳＧという）を取得する。第５行は、また、ＳＳＴＧパス内の終了ノードへの各遷移に対応するＲＴＬ回路内のカットを識別する。カットとは、本明細書においては、状態遷移によりある状態から別の状態へ伝搬する変数のセットとして定義される。図７の疑似コードの第６行は、ＳＳＴＧパスを列挙するのに必要なすべての状態遷移判定をカプセル化するパスシグナル(Pathsignal)という信号を生成する。ＳＳＴＧパスに対応するＢＳＴＧ内のパスを識別するシンボリックシミュレーションは、図７の疑似コードの第７行で、手続きConstrained#symbolic#simulation()によって、ＳＳＴＧパスのPathsignalを用いて実行される。

【０１１６】図８を参照しながら、Constrained#symbolic#simulation()の詳細について説明する。ＢＳＴＧ内のルート状態からはじめて、そのタスクは、Pathsignalと両立する遷移により到達可能な状態を識別することである。到達した各状態で、対応する信号を識別するために非解釈シンボリックシミュレーションを実行する（図８の第５行）。次に、その状態からの、Pathsignalと両立する出遷移を識別する。このプロセスは、ＢＳＴＧ内のＥＮＤ状態に到達するまで続く。

【０１１７】図７の全体アルゴリズムに戻って、次のステップは、列挙されたパス内のループから不変項を抽出することである（図７の第８～１２行）。このステップは、ループが存在しないときには不要となる。パスに沿って遭遇する各ＳＣＣに対して、図９に記載した手続きreturn#loop#invariants()を呼び出す。この手続きは、変数対応セットをcorresp#set#listとして返す。このリストのうち、前のシンボリックシミュレーションの結果としてＳＣＣカットで得られた変数対応より小さい各対応セットに対して、図７の第１２行に示すように、ＳＣＣに続くパス部分のシンボリックシミュレーションを再実行しなければならない。図７の第１３行及び第１４行は、得られた出力等価性が、パス条件以外の条件付きであるかどうかをテストする。そのように条件付きである場合、ＳＴＧは等価でないと見なされる。出力が、列挙されたすべてのパスに対して無条件に等価である場合、ＳＴＧは等価であると見なされる。

【０１１８】図９を参照すると、return#loop#invariants()への入力は、列挙されたパスにおいてＳＣＣの終了ノードに遭遇する３つのインスタンスに対応するＳＳＧ内の３個のカット（ssg#cuts 1, 2, and 3）である。この手続きは、ループに続くパス部分のシンボリックシミュレーションが実行されなければならないような対応セットのリストを返す。図９の手続きの第１行は、ssg#cutに対応するＢＳＧにおける変数（bsg#cutsという）を取得する。第２行及び第３行は、ＢＳＧにおいて導出されたカットどうしの間の２つの回路を取得し、カット２とカット３の間の回路が、カット１とカット２の間の回路の単なる別のインスタンス（コピー）であるかどうかを確かめる。そうでない場合、対応がないと見なされ、適当なcorresp#set#listが返される。カット間の回路が同型である場合、非自明な対応セットが存在する可能性がある。

【０１１９】このセットを見つけるため、手続きは、最初にカット２からはじめて、一度に１回のループ実行（すなわち、２つのカットの間の部分）だけ進むように、ＳＳＧとＢＳＧをシンボリックシミュレートする。各シミュレーションの最後に得られる変数対応（corresp#set$_{n+1}$）を、そのシミュレーションの最初における対応（corresp#set$_n$）と比較する。これらのセットが同一である場合、これは要求された固定点であり、手続きは、この点で見出した対応リストをcorresp#set#listの一部として返す。そうでない場合、corresp#set$_{n+1}$を初期変数対応として、１ループ実行のシンボリックシミュレーションを繰り返す。この手続きはまた、１回の実行のシンボリックシミュレーションにおける対応セットの使用により新たな変数対応が生成されたときには、この対応セットをcorresp#set#listに追加する。これは、例４及び例５において議論したようなフォールスポジティブを避けるためである。

【０１２０】すべての変数対応を識別するのに要する反復回数は、可能な変数対応の総数によって制限される。最悪の場合、これは、ＳＳＧとＢＳＧ内のループ本体の変数の個数の積になりうる。実際には、変数対応の数は変数の個数に関して線形であり、ほとんどの対応は、最初の実行自体の後に見出される。従って、この手続きは、有限回の（実際には、非常に少ない）ループ反復でループ不変項を得る手段である。

【０１２１】４．２．３　アルゴリズムの正当性及び有効範囲

フォールスポジティブは、２つの表現が実際には等価でないときに、検証ツールがそれらを等価であるとみなす場合に生じる。フォールスネガティブは、２つの表現が

実際には等価であるときに、検証ツールがそれらを等価でないとみなすときに生じる。次の定理は、本発明のアルゴリズムを特徴づける。

【０１２２】定理２：図７の手続きCompare#STGsは、（ａ）「代表的」スケジューリング、及び、（ｂ）実現不可能な反復カウントによるネガティブの可能性がない、という仮定の下で、フォールスポジティブまたはフォールスネガティブを発生しないことが保証される。

【０１２３】（証明）スケジュール及びビヘイビアがいずれも非巡回的であるとき、フォールスポジティブが発生しないことは、基本的なシンボリックシミュレーションに基づく等価性チェッカの性質である。フォールスネガティブは、非巡回的である場合、シンボリックシミュレータによって非解釈とされる演算の機能の知識が最適化で使用されるときにのみ発生しうる。残りの解析では、シンボリックシミュレーションに基づく等価性チェッカは、非巡回パスにおいて正しい変数対応を見出すという事実に依拠することができる。

【０１２４】さらに興味深いことは、ループが存在する場合に本発明のアルゴリズムでいつフォールスネガティブ及びポジティブが発生し得るかの解析である。ビヘイビア記述は巡回的（ループを含む）であるがスケジュール記述は非巡回的（ループを含まない）である場合、あるいはその逆の場合は、代表的スケジューリングによって許容されない。両方の記述にループがある場合、生成される変数対応が多すぎるときにフォールスポジティブが起こり、生成される変数対応が少なすぎるときにフォールスネガティブが起こる。

【０１２５】まずフォールスネガティブを考える。ループ停止の正当性は、本発明の手続きでは、スケジュール及びビヘイビアにおけるループの停止条件どうしの間の対応を確定することによってチェックされる。本発明のアプローチは、実現不可能な反復カウントについて知らない。スケジュール記述を生成するために使用される最適化が実現不可能な反復カウントの知識を使用する場合、本発明の手続きはフォールスネガティブを報告する可能性がある。また、実現不可能な回数の反復の後にのみループ本体どうしの間の差が「活性化」されるときにもこれは起こり得る。従って、代表的スケジューリングであり、かつ、実現可能な反復カウントがないという仮定の下では、このようなフォールスネガティブは起こり得ない（証明終）。

【０１２６】図９に示すように、不変項抽出手続きは、反復のたびに変化しない変数対応のセットを識別するまで、ループを収束するまで反復する。このプロセスで生成される各極小変数対応セット（これは、他の対応セットのサブセットではない）は、ループ終了点の後のコードのシミュレーションを実行するために別々に使用される。ループのｎ回の実行後に得られる変数対応セットをCS_nで表す。以後のシミュレーションで使用される、極小変数対応セットの集合を$\{CS^i\}$で表す。すなわち、$CS_n \subseteq \{CS^i\}$は、極小対応セットであり、ループ終了以後のシンボリックシミュレーションのために使用される。

【０１２７】明らかに、$CS_n \subseteq \{CS^i\}$でのシミュレーションにより生じるネガティブの等価性結果は、実現不可能な反復カウントがないという仮定と、シンボリックシミュレーション手続きの基本的性質により、真のネガティブである。

【０１２８】次に、フォールスポジティブを考える。変数対応のセットの固定点（すなわち、$CS_k = CS_{k+1}$）に到達するのにｋ＋１回の反復が必要であると仮定する。帰納法により、このセットは、$n \geq k$に対して、ｎ回の反復に対応する実行されたパスに対する正しい変数対応のセットであるということができる。従って、CS_kでのシミュレーションの後に得られるポジティブの等価性結果は、$n \geq k$のすべてのｎに対して真のポジティブである。

【０１２９】ここで、スケジュールにおけるｋより少ない回数のループの実行に対応するパスを考える。すなわち、$n < k$とする。アルゴリズムは、ｎ回の反復後の終了をチェックするように正しく動作すること、すなわち、この場合にフォールスポジティブがないことを示す必要がある。考慮すべき次の２つの場合がある。

【０１３０】１．　CS_nは、極小対応セットのうちの１つである。すなわち、$CS_n \subseteq CS^i$である。この場合、すべての極小対応セットは、ループ終了点以後明示的にシミュレートされるため、CS_nはフォールスポジティブを発生し得ない。

【０１３１】２．　CS_nは、極小対応セットのうちの１つでない。極小対応セットの定義により、CS_nは、$\{CS^i\}$内の対応セットのうちの１つのスーパーセットでなければならない。この場合、$\{CS^i\}$内のすべての対応セットでのシンボリックシミュレーションがポジティブの結果を発生する場合、CS_nでのシンボリックシミュレーションも同様となり、CS_nで別個のシミュレーションをする必要はない。（他方、$\{CS^i\}$内のいずれかの対応セットでのシンボリックシミュレーションがネガティブを発生する場合、記述は非等価であり、CS_nでのシンボリックシミュレーションは意味がない。）

【０１３２】［４．２．３．１　ネストしたループの扱い］ネストしたループを扱うためには、内側のループに入るたびに不変項を解析しなければならない。前述のCompare#STGs手続きに加えて、ループネスティングを決定する解析が必要となる。ネストしたループをどのように扱うかについての直観的な説明は、セクション５．２のネストしたループの例のケーススタディを参照。

【０１３３】［４．３　アルゴリズムの効率］アルゴリズムの効率は、基本的に、次の３つのファクタから導き

出される。
（１）データパス状態は列挙されない。
（２）算術は解釈されない。
（３）不変項を抽出するためにループは完了まで反復されない。
【０１３４】ファクタ（１）及び（２）は、等価性チェックのためのアルゴリズムの内側ループで使用される非解釈シンボリックシミュレータに含まれる。ファクタ（３）は、本発明の不変項抽出アルゴリズムによる。手続きCompare#STGs()におけるＳＳＴＧ内のＳＣＣ識別及び終了点の識別は、ＳＳＴＧのサイズに関して線形である。Compare#STGs()におけるパス列挙は、つぶしたＳＳＴＧに対して行われる。これは、最悪の場合に列挙されるパスの個数が、ＳＳＴＧ内の状態数ではなくスケジュール内のループの個数に関して指数関数的になることを意味する。このパス数は一般に非常に小さい可能性が高い。不変項を抽出するために、列挙に要するループ反復回数は、可能な変数対応関係の個数によって制限される。最悪の場合、これは、ループ本体内の変数の個数に関して２次になり得る。実際には、スケジューリングにおけるエラーがないときの代表的スケジュールの場合、すべての変数対応は、２回のループ反復の列挙により見出される。シンボリックシミュレーション中のブール演算には二分決定ダイヤグラム（ＢＤＤ）が必要とされるが、このようなサブ回路は実際は非常に小さいため、ＢＤＤ生成がボトルネックとなることはない。シンボリックモデルチェック(symbolic model checking)のような技術に比べて、本発明のアルゴリズムのランタイム計算量（複雑さ）は小さいため、本発明のアルゴリズムは、取り組んでいる特定の検証問題に対する高速なカスタマイズされた解法として適している。
【０１３５】［４．４　スケジューリング検証システム］本発明の重要な特徴は、回路のスケジュールがビヘイビア記述から得られるような回路のスケジューリングの正当性をチェックするシステムとして実現される。このようなシステムの好ましい実施例を図１５に示す。ループ不変項抽出器１４．１は、ループが存在するときに非巡回スレッドの十分なセットを決定する。シンボリックシミュレータ１４．２は、ループ不変項を抽出する。等価性証明器１４．３は、非巡回スレッドの等価性を証明する。このシステムは、
・サイクル境界の導入
・演算並べ替え
・ループの展開、巻付け、折畳み及びパイプライン化
・演算の投機実行
のうちの１つ以上により変換されたビヘイビア記述を扱うことが可能である。
【０１３６】本発明のもう１つの重要な特徴は、回路のビヘイビア記述に対して回路のスケジュールを検証するシステムとして実現される。このようなシステムの好ましい実施例を図１６に示す。スケジュール状態遷移グラフ生成器１５．２は、１５．１からスケジュールを受け取り、スケジュールをスケジュール状態遷移グラフとして指定する。ビヘイビア状態遷移グラフ生成器１５．３は、回路のビヘイビアをビヘイビア状態遷移グラフとして指定する。スケジュールスレッドセレクタ１５．４は、スケジュールを受け取り、スケジュール状態遷移グラフから実行のスケジュールスレッドを選択する。ビヘイビアスレッドセレクタ１５．５は、ビヘイビアを受け取り、ビヘイビア状態遷移グラフから対応するビヘイビアスレッドを選択する。変換器１５．６は、スケジュールスレッドをスケジュール構造グラフに、及び、ビヘイビアスレッドをビヘイビア構造グラフに変換する。等価性チェッカ１５．７は、前記スケジュール構造グラフと前記ビヘイビア構造グラフの等価性をチェックする。
【０１３７】［４．５　スケジューリング検証コンピュータシステム］コンピュータは、本発明の技術を実現するための非常に有効な手段である。本発明の技術を実現するこのようなコンピュータシステムもまた本発明の技術的範囲内に入る。このようなコンピュータは、プロセッサ及びメモリを有する。メモリは、コンピュータが回路のスケジューリングの正当性をチェックすることを可能にする命令を含む。ここで、回路のスケジュールは、そのビヘイビア記述から得られる。具体的には、メモリ内の命令は、ループが存在するときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出する命令を含む。さらに、命令は、ループ不変項を抽出するためのシンボリックシミュレーションの命令を含む。さらに、命令は、非巡回スレッドの等価性を証明する命令を含む。
【０１３８】なお、コンピュータは、ＰＣ、メインフレーム、ワークステーションあるいはネットワーク上のリモートコンピュータを含むいかなる種類のコンピュータとすることも可能である。
【０１３９】コンピュータシステムの好ましい実施例は、命令を含むメモリを有するコンピュータからなる。この命令は、コンピュータが、図４に示した擬似コードを実行することを可能にする。別の好ましい実施例は、コンピュータが、図７～図９に示した擬似コードを、単独に、またはすべての可能な組合せで、実行することを可能にする命令を含むメモリを有するコンピュータからなる。
【０１４０】なお、命令は、高水準言語、低水準言語、アセンブリ言語及び機械語を含む（これらに限定されない）任意の形式とすることが可能である。
【０１４１】［４．６　スケジューリング検証コンピュータプログラム製品］本発明の重要な特徴は、コンピュータプログラム製品として実現される。このプログラム製品は、コンピュータが回路のスケジューリングの正当性をチェックすることを可能にする命令を有するコンピ

ュータ可読媒体を含む。なお、コンピュータ可読媒体は、フロッピー（登録商標）ディスク、ハードディスク、ＣＤ、チップ、テープ、ＩＣ付きカートリッジなどを含む（これらに限定されない）任意の固定媒体を含む。コンピュータ可読媒体は、ネットワークを通じて伝送される、あるいは、インターネットからダウンロードされる命令も含む。

【０１４２】好ましい実施例では、コンピュータコードは、コンピュータが回路のスケジューリングの正当性をチェックすることを可能にする。ここで、回路に対するスケジュールは、ビヘイビア記述から得られる。コンピュータコードは、ループが存在するときに非巡回スレッドの十分なセットを決定するためにループ不変項を抽出するコンピュータコードと、ループ不変項を抽出するためのシンボリックシミュレーションのコンピュータコードと、非巡回スレッドの等価性を証明するコンピュータコードとを含む。

【０１４３】コンピュータプログラム製品の好ましい実施例は、コンピュータコードを含むコンピュータ可読媒体を含む。このコンピュータコードは、コンピュータが、図４に示した擬似コードを実行することを可能にする。別の好ましい実施例は、コンピュータが、図７～図９に示した擬似コードを、単独に、またはすべての可能な組合せで、実行することを可能にするコードを含むコンピュータ可読媒体を含むコンピュータプログラム製品からなる。

【０１４４】なお、コンピュータコードは、高水準言語、低水準言語、アセンブリ言語及び機械語を含む（これらに限定されない）任意の形式とすることが可能である。

【０１４５】

【発明の効果】［４．７　結果：ケーススタディ］本発明を適用した結果について、本発明のアルゴリズムを我々の実際のスケジューリングの例に適用した詳細なケーススタディの形で提示する。これらの設計は、状態変数の個数及び算術演算の計算量（複雑さ）に関して十分に大きいため、状態マシン等価性やシンボリックモデルチェックに基づく従来のＢＤＤによる検証アプローチでは確実に失敗する。

【０１４６】４．７．１　投機スケジューリングの例

図１０（ａ）に示すビヘイビアＳＴＧを考える。このビヘイビアを、投機実行及びループ変換を含む最新のスケジューラによってスケジューリングした。結果として得られたスケジュールＳＴＧを図１０（ｂ）に示す。ビヘイビア内のループを考える。定常状態では（すなわち、ループが多数回実行されると仮定すると）、スケジュールＳＴＧは、状態Ｓ６からなるＳＣＣをたどることに注意する。スケジュールの解析を実行することによって、定常状態では、ループの新たな反復がクロックサイクルごとに開始され、大きなパフォーマンス改善につながることを示すことが可能である。

【０１４７】検証の観点から、この例で興味深い点は、この例は、セクション１．２で言及したほとんどの最適化（サイクル境界の導入、演算の並べ替え、パスセグメントの複製、ループパイプライン化、及び投機実行）を同時に含むことである。これらの最適化は、スケジュールの複雑さを大幅に増大させる。ビヘイビアＳＴＧのＶＨＤＬ記述は、１２２行のコードからなり、７個の演算及び８個の中間変数（プライマリ入力及びプライマリ出力を除く）を含むのに対して、スケジュールＳＴＧのＶＨＤＬ記述は、２８９行のコードからなり、４７個の演算及び５４個の中間変数を含む。明らかに、ビヘイビアとスケジュールの構造同型チェックでは、これらの等価性を証明することはできない。（データパス＋制御）状態空間をたどるＶＳＩのような従来の低レベル（例えば、ゲートレベル）のＦＳＭ等価性チェックツールでは非常に困難になる。ビヘイビアＳＴＧは２５０個の状態ビットを含み、スケジュールＳＴＧはさらに大幅に多くの状態ビットを含む。R. K. Brayton et al., "VIS: A system for verification and synthesis", in Proc. Int. Conf. Computer-Aided Verification, July 1996、を参照。

【０１４８】図１０（ａ）及び（ｂ）に示す２つのＳＴＧの等価性を証明するために本発明の検証手続きで実行されるさまざまなステップについて説明する。図７に示した手続きCompare#STGsの第４行により、ＳＳＴＧで最初に列挙されるスレッドは、（ＳＡ，ＳＧ）、（ＳＡ，ＳＢ，ＳＣ，ＳＤ，ＳＥ，ＳＦ，ＳＧ）、（ＳＡ，ＳＢ，ＳＣ，ＳＤ，ＳＥ，ＳＦ，ＳＦ，ＳＧ）、及び（ＳＡ，ＳＢ，ＳＣ，ＳＤ，ＳＥ，ＳＦ，ＳＦ，ＳＦ，ＳＧ）である。これらのスレッドのそれぞれについて、手続きは、ビヘイビア及びスケジュールに対する構造グラフを生成し、セクション３で説明したシンボリックシミュレーション手続きを用いてその出力の等価性を証明する。さらに、スレッド（ＳＡ，ＳＢ，ＳＣ，ＳＤ，ＳＥ，ＳＦ，ＳＦ，ＳＧ）については、ループ不変項を抽出する手続きを呼び出す。

【０１４９】シンボリックシミュレーションがどのように進行してループ不変項を検出するかを調べるために、ここでは、スレッドＴ１＝（ＳＡ，ＳＢ，ＳＣ，ＳＤ，ＳＥ，ＳＦ，ＳＦ，ＳＧ）を考える。スケジュール構造グラフSSG_{T1}を図１１に示す。スケジュール内の状態境界に対応する構造グラフ内のカットは点線を用いて示される。

【０１５０】手続きConstrained#Symbolic#Simulationは、ビヘイビア、及び対応する構造グラフにおける、対応するスレッドを自動的に抽出する。結果として得られるビヘイビアスレッドはＴ２＝（Ｓ０，Ｓ１，...，Ｓ７，Ｓ１，...，Ｓ７，Ｓ１，Ｓ８）であり、その構造グラフ（BSG_{T2}）は図１２に示される。なお、スケジ

ュールとビヘイビアのＳＴＧにおける状態境界は対応していないため、SSG_{T1}内のカットに対応するBSG_{T2}内の「カット」は、等価関係を用いて決定される。例えば、SSG_{T1}内の第１のカットにおける信号に関する等価関係は、(t4, t4, 1), (M2, M2, 1), (c, c, 1), (i#gt1#0, i#1, 1), (i', i#1, c)である。ここで、エントリ（ｓ１，ｓ２，cond）は、SSG_{T1}内の信号ｓ１とBSG_{T2}内の信号ｓ２が条件condの下で等価であることを意味する。理解されるように、信号cとi#1は、BSG_{T2}内の対応するカットを形成する。SSG_{T1}の第５及び第６のカットに対応してBSG_{T2}内で形成されるカットは、図１２で点線を用いて示されている。これらのカットは、状態ＳＦを含むスケジュールＳＴＧループの第１回の実行の開始及び終了を表す。これらの２つのカットのメンバである信号に関する等価関係は、(i', i#1, c), (i#gt1#1, i#2, c), (t4, t4, c), (t3#gt1#1, t3#1, c)及び(c', c#1, c), (M2', M2#1, c), (t4', t4#1, c.c#1, (i", i#2, c.c#1)である。

【０１５１】上記のことから明らかに示されるように、ループ境界変数の多くについての対応は存在しない（例えば、t3'#gt1#1、t2'#gt1#2、など）。さらに解析すれば理解されるように、SSG_{T1}の大部分（図中影を付けた部分）はシミュレートされていない。スケジュール内のループをもう１回展開することにより、影部分の変数について新たな等価関係が見出される（すなわち、ループ不変項のセットが増大する）。この例では、ループ変数間の等価関係が収束するためには全部で６回ループを展開する必要があることを示すことができる。なお、ビヘイビア内のループは、スケジュールを導出する際にスケジューラによって（偶然の一致ではなく）ファクタ６でパイプライン化されたことに注目すると興味深い。

【０１５２】４．７．２　Ｘ．２５通信プロトコル

この例は、Ｘ．２５通信プロトコルの送信(send)プロセスである。S. Bhattacharya, S. Dey, and F. Brglez, "Performance analysis and optimization ofschedules for conditional and loop-intensive specifications", in Proc. Design Automation Conf., pp.491-496, June 1994、を参照。演算への状態の直接的な割当てによるビヘイビアに対する制御フローグラフを図１３（ａ）に示す。なお、これは配列変数を使用している。配列アクセスは、非解釈関数であるとみなされる。配列インデックスと配列名は対応することが保証される。これから得られる正しいスケジュールを図１３（ｂ）に示す。各状態内の数字は、その状態内で実行される演算に対応する。この例は２つの理由により興味深い。第１の理由は、状態Ｓ１１及びＳ１２に対応するネストしたループである。第２の理由は、スケジュール内のパスに対して生成されるデータフローグラフは、ビヘイビアにおいて生成される対応するパスと構造的に同型ではないことである。図１３（ｃ）に、同じビヘイビアに対する正しくないスケジュールを示す。ループのネスティングを見出すためには、スケジュールのＳＴＧに対する正規表現(regularexpression)を導出する。Z. Kohavi, "Switching and Finite Automata Theory", McGraw-Hill Book Company, second ed., 1978、を参照。

【０１５３】まず図１３（ｂ）を考える。これに対して導出される正規表現は$S_0S_1S_2S_4(S_3S_4)^*(S_5S_2S_4(S_3S_4)^*)^*S_6$である。上付きの＊のある各部分表現はループ本体を構成する。これは明確にループのネスティングを識別する。パス列挙がループ本体に遭遇するたびに、そのループに対する不変項を識別しなければならない。例えば、部分表現$(S_5S_2S_4(S_3S_4)^*)^*$において、内側ループ$(S_3S_4)^*$の不変項は、外側ループの不変項を導出するために外側ループに要求される反復回数と同じ回数だけ抽出されなければならない。手続きの残りの部分は前と同じであり、第１のケーススタディの通りに従う。

【０１５４】次に、図１３（ｃ）を考える。導出される正規表現は、$S_0S_1S_2S_4((S_3S_2S_4)^*+(S_5S_2S_4)^*)^*S_6$である。この正規表現は、外側ループの内部に共通の部分表現を有する２個のループを含むという点で前のものとは異なる。部分表現$((S_3S_2S_4)^*+(S_5S_2S_4)^*)^*$における、外側ループのループ不変項を決定することは興味深い。これを達成するため、変数対応が安定するまで、外側ループの有限回の反復が評価されなければならない。外側ループが反復されるたびに、内側ループのインスタンス化のすべての可能な組合せが考慮される。例えば、外側ループが部分表現$(A^*+B^*)^*$である場合、外側ループの２回の実行は、４個のパス、すなわち、

A^*A^*, A^*B^*, B^*A^*, B^*B^*

に沿って変数対応を計算しなければならないことを意味する。なお、各パスは、それぞれネスティングのないループからなる。外側ループの３回目の実行で、列挙されるパスは次のようになる。

$A^*A^*A^*$, $A^*B^*A^*$, $B^*A^*A^*$, $B^*B^*A^*$, $A^*A^*B^*$, $A^*B^*B^*$, $B^*A^*B^*$, $B^*B^*B^*$

外側ループに対するループ不変性を証明するためには、外側ループのｎ回目の実行における部分パスP_nから生成される変数対応が、P_nから導出されるｎ＋１回目の実行におけるすべての部分パスから生成される対応と同じままであることを示さなければならない。例えば、上記の仮設的な例では、外側ループの２回の実行後にA^*A^*から生成されるループ対応は、３回目の実行において$A^*A^*A^*$及び$A^*A^*B^*$から生成される対応と同じままでなければならない。図１３（ｃ）におけるスケジュールは、状態Ｓ３から出る遷移の正しくない実現の結果、実際には正しくない。これは、本発明の手続きでは検出される。２つの表現におけるdata変数の間に対応を確立することが不可能であるからである。

【図１】

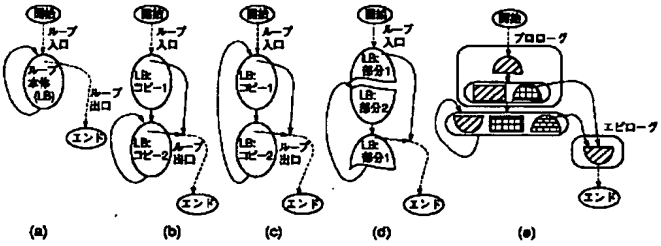
```
P1  :  process
     Variable  t6 :  integer;

begin
  x_var <= Xinport;
  dx_var <= Dxport
  y_var  <= Yinport;
  u_var <= Uinport;
  wait for 0ns;
  wait until clk='1' and clk'event; -- CLOCK EDGE

  while (x_var < a_var) loop
    t6 := u_var - u_var * dx_var * x_var;
    u_var <= t6 - dx_var * three * y_var;
    wait until clk='1' and clk'event; -- CLOCK EDGE
    y_var <= y_var + u_var * dx_var;
    x_var <= x_var + dx_var;
    wait for 0ns;
  end loop;

  Xoutport <= x_var;
  Youtport <= y_var;
  Uoutport <= u_var;
end process P1;
```

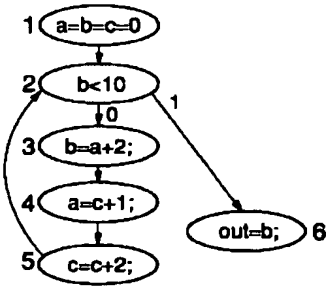
【図３】



【図６】

```
p=0; q=0; r=0;
while (q<10)   {
    q=p+1;
    p=r+1;
    r=r+1;
}
out=q;
```

(a) ビヘイビアコード



(b) スケジュール STG

【図４】

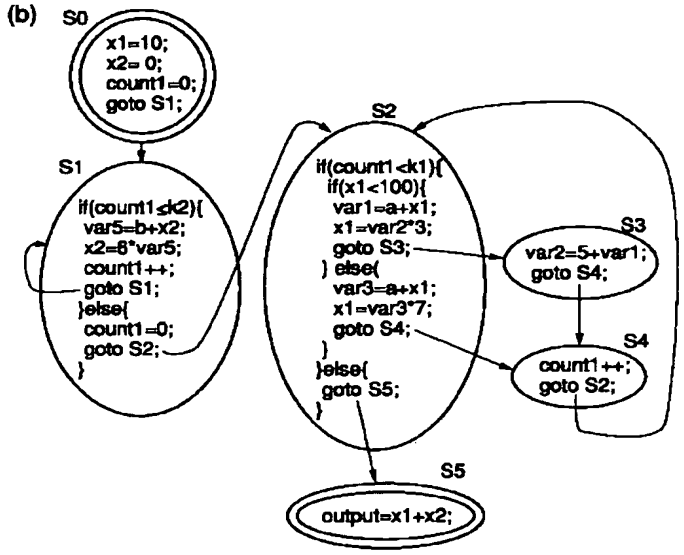
```
Procedure COMPARE_STRUCTURE_GRAPHS(SSG_T, BSG_T')
  Arr1 := DFS_SORT(BSG_T')
  Arr2 := DFS_SORT(SSG_T)
  Identify basis variables in BSG_T';
  Symbolic simulate BSG_T' to express non-basis vars in
    terms of basis vars;
  Construct equivalence lists for IN nodes in SSG_T;
  For_each_element(Arr2, v) {
    switch(TYPE(v)) {
    case MUX:
      For each data input v_fanin of v {
        For each entry (u,c) in equivalence list of v_fanin
          ADD_EQUIVALENCE(v, u, c ⋂ select_cond);
    case OP:
      Identify corresponding OP vertex v_op in BSG_T' such that
        (i) v_op and v perform the same operation, AND
        (ii) inputs of v have conditional equivalences
        with corresponding inputs of v_op;
      cond = conjunction of conditions;
      If cond ≠ 0 {
        ADD_EQUIVALENCE(v, v_op, cond); }
    case LOGIC:
      convert input lists into BDD nodes and propagate;
    case PO:
      If equivalence exists with corresponding PO in BSG_T'
      and condition is 1
        continue;
      else
        return(Error); } }
  return(Equivalent);
```

【図２】

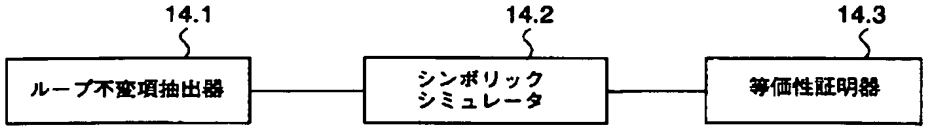
(a)

```
int Test1(int k1, int k2, int a, int b){
      int x1, x2, var1, var2, var3, var4, var5;
      x1 = 10;
      x2 = 0;
      for (int count1 = 0; count1 <= k1; count1++) {
      // <=1 and ++1
            if (x1 < 100) {          // +1
                  var1 = a + x1;     // +1
                  var2 = 5 + var1;   // +2
                  x1 = var2 * 3;     // *1
            }
            else {
                  var3 = a + x1;     // +3
                  x1 = var3 * 7;     // *2
            }
      }
      for (int count1 = 0; count1 <= k2; count1++) {
      // <=2 and ++2
            var5 = b + x2;     // +4
            x2 = 6 * var5;     // *3
      }
      return (x1 + x2);        // +5
}
```

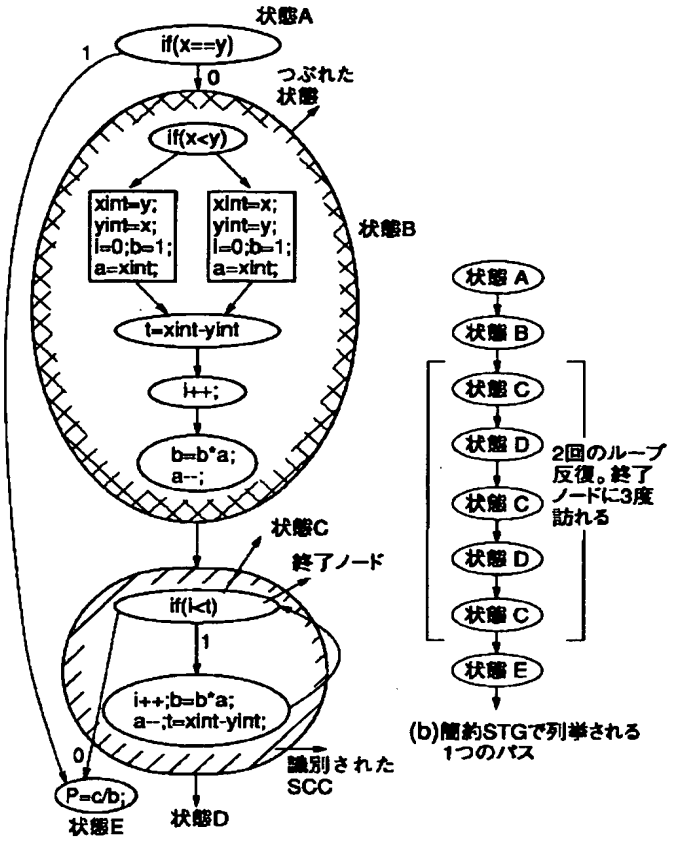
(b)



【図１５】



【図５】



(a)SSTGから得られる簡約グラフ

【図８】

```
Procedure CONSTRAINED_SYMBOLIC_SIMULATION(BSTG, SSG,
   Patheignal)
{
1. Permissible_Paths = Begin state in BSTG
2. do {
3.   Foreach unvisited state S_i in Permissible_Paths {
4.     BSG = Generate_Structural_RTL(BSTG,
         Permissible_Paths)
5.     Unintepreted_Symbolic_Simulation(SSG, BSG)
6.     Foreach outgoing transition S_i --> S_j {
7.       If conjunction of transition condition and Patheignal
           is not 0 {
8.         Add new copy S_j to Permissible_Paths
         }
       }
     }
9. } until (the only unvisited states in Permissible_Paths
   are instances of the END state)
}
```

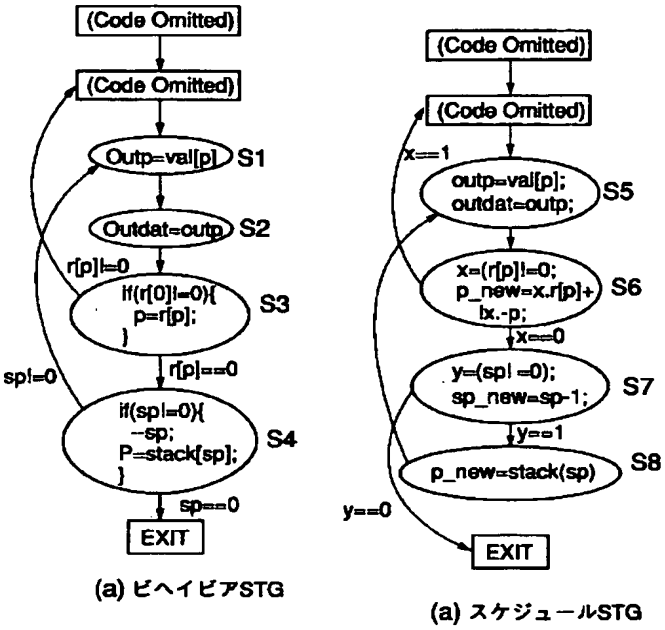
【図９】

```
Procedure RETURN_LOOP_INVARIANTS(seg_cuts 1, 2 and 3)
{
1. bsg_cuts = cuts in BSG corresponding to seg_cuts
2. Derive the two subcircuits between bsg_cuts 1, 2 and 3
3. Check that these subcircuits are isomorphic
4. If not isomorphic, return corresp_set_list = {NULL}
5. corresp_set_n = correspondences associated with bsg_cut_2
6. corresp_set_n+1 = correspondences associated with bsg_cut_3
7. while (corresp_set_n ≠ corresp_set_n+1) {
8.   if corresp_set_n+1 is a subset of corresp_set_n {
9.     corresp_set_list = corresp_set_list ∪ corresp_set_n+1
     }
10.  corresp_set_n = corresp_set_n+1
11.  propagate corresp_set_n forward for one loop iteration
       by symbolic simulation
   }
12.corresp_set_list = corresp_set_list ∪ corresp_set_n+1
13.remove all entries in corresp_set_list that are supersets of
     other entries;
14.return corresp_set_list
}
```

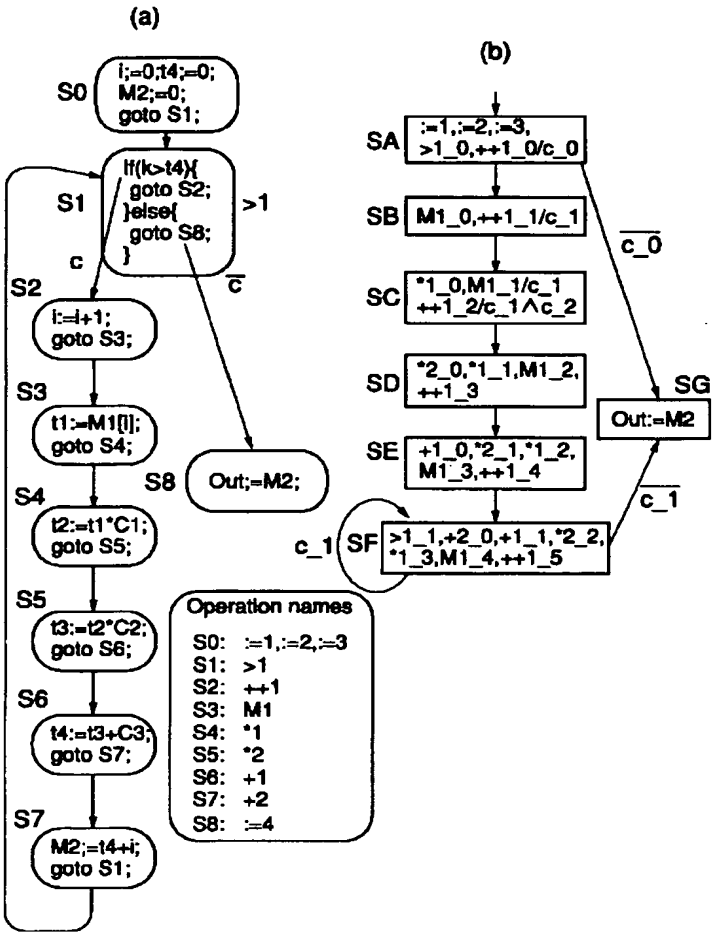
【図７】

```
Procedure COMPARE_STGs(BSTG, SSTG)
{
1. Identify SCCs in SSTG
2. Identify exit nodes in each SCC
3. Collapse nodes not in SCCs and
   nodes in SCCs that are not exit nodes
4. Foreach path from root to sink such that
     any exit point appears 0, 1, and 3 times {
# Through symbolic simulation,
# identify the corresponding path in the BSTG
5.   Obtain the structural RTL circuit (SSG) for the path -
     also returns cuts for the begin and end of each SCC
6.   Add circuitry to SSG to generate Pathsignal
7.   Constrained_symbolic_simulation(BSTG, SSG, Pathsignal)
8.   Foreach SCC in the path in order from root {
9.     corresp_set_list = return_loop_invariants(ssg_cuts 1, 2 and 3)
10.    Foreach corresp_set in corresp_set_list {
11.      If (corresp_set is smaller than the correspondence set
         associated with ssg_cut 1) {
12.        Redo symbolic simulation from the third cut
           using only the corresp_set
         }
13.      Cofactor output equivalence conditions with path condition
14.      If resulting output equivalence is not unconditional {
15.        return not equivalent
         }
       }
     }
   }
16.return equivalent
}
```

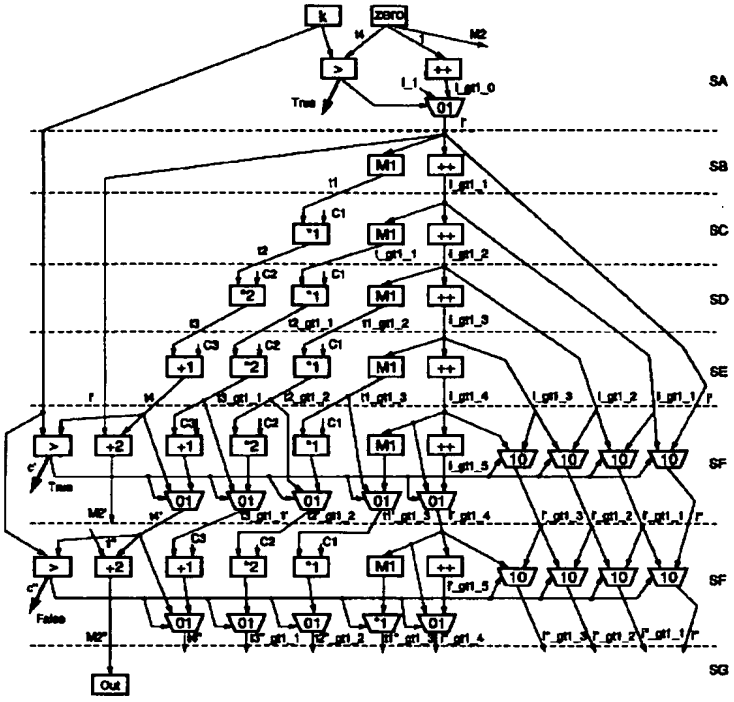
【図１４】



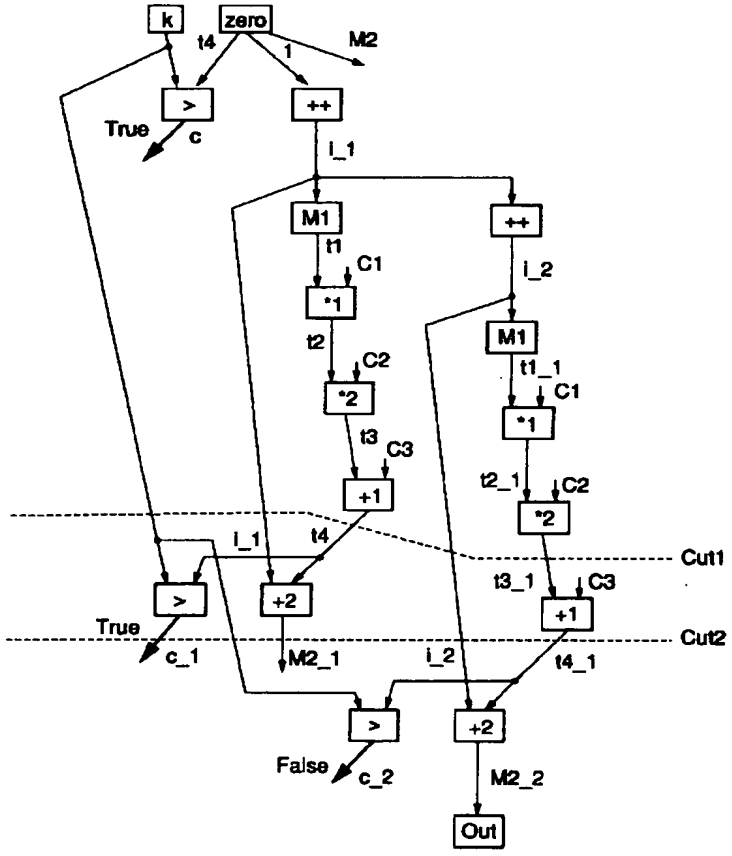
【図１０】


(a)
S0
i;=0;t4;=0;
M2;=0;
goto S1;
S1
If(k>t4){
goto S2;
}else{
goto S8;
}
>1
c
c̄
S2
i:=i+1;
goto S3;
S3
t1:=M1[i];
goto S4;
S8
Out;=M2;
S4
t2:=t1*C1;
goto S5;
S5
t3:=t2*C2;
goto S6;
S6
t4:=t3+C3;
goto S7;
S7
M2;=t4+i;
goto S1;
Operation names
S0: :=1,:=2,:=3
S1: >1
S2: ++1
S3: M1
S4: *1
S5: *2
S6: +1
S7: +2
S8: :=4
(b)
SA
:=1,:=2,:=3,
>1_0,++1_0/c_0
SB
M1_0,++1_1/c_1
SC
*1_0,M1_1/c_1
++1_2/c_1∧c_2
SD
*2_0,*1_1,M1_2,
++1_3
SE
+1_0,*2_1,*1_2,
M1_3,++1_4
SF
>1_1,+2_0,+1_1,*2_2,
*1_3,M1_4,++1_5
c_1
c_0̄
SG
Out:=M2
c_1̄

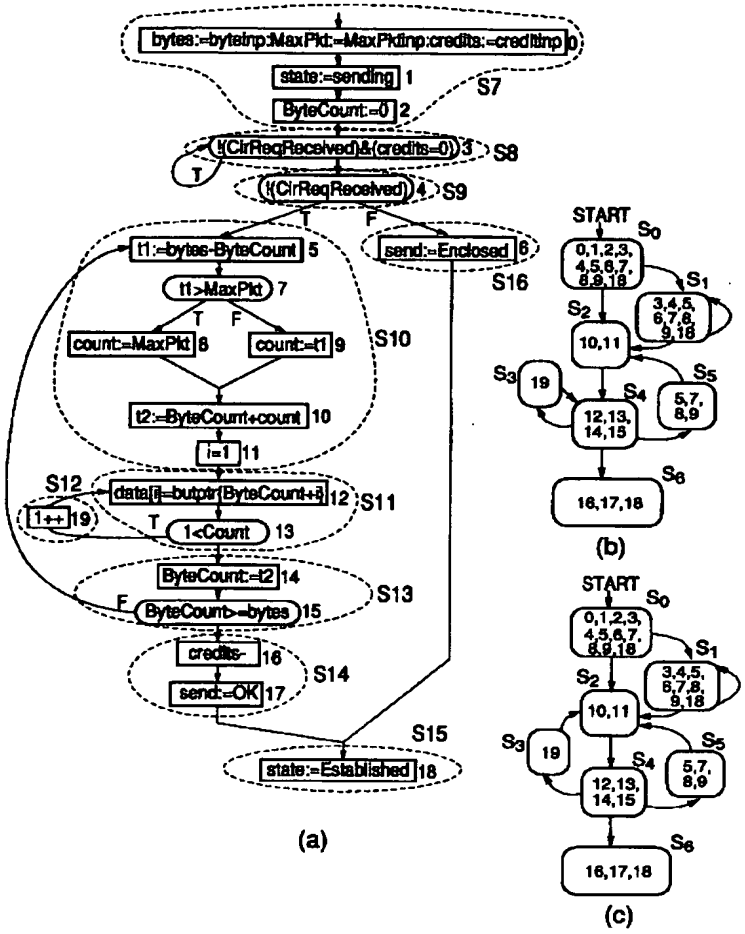
【図１１】


k
zero
M2
>
++
True
01
SA
M1
SB
*1
C1
SC
*2
C2
SD
+1
C3
SE
+2
SF
Out
SG

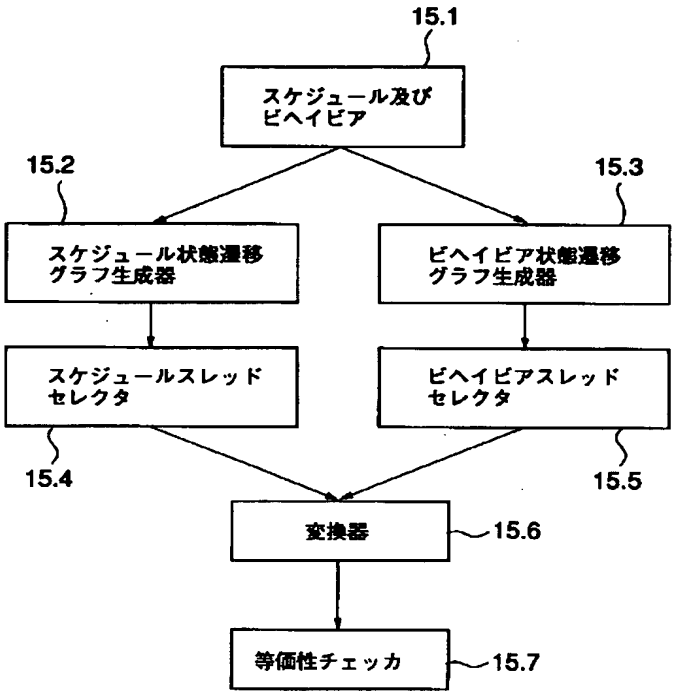
【図１２】


k
zero
t4
1
M2
>
True
c
++
I_1
M1
t1
C1
*1
t2
C2
*2
t3
C3
+1
I_2
t1_1
t2_1
t3_1
Cut1
Cut2
+2
c_1
M2_1
t4_1
False
c_2
M2_2
Out

【図１３】


bytes:=bytesinp;MaxPkt:=MaxPktinp;credits:=creditinp 0
state:=sending 1
ByteCount:=0 2
S7
!(ClrReqReceived)&(credits=0) 3
S8
T
!(ClrReqReceived) 4
S9
T
F
t1:=bytes-ByteCount 5
send:=Enclosed 6
S16
t1>MaxPkt 7
T
F
count:=MaxPkt 8
count:=t1 9
S10
t2:=ByteCount+count 10
i=1 11
S12
data[i]=outptr[ByteCount+i] 12
S11
i++ 19
T
i<Count 13
ByteCount:=t2 14
S13
F
ByteCount>=bytes 15
credits- 16
S14
send:=OK 17
S15
state:=Established 18
(a)
START
S0
0,1,2,3,4,5,6,7,8,9,18
S1
3,4,5,6,7,8,9,18
S2
10,11
S3
19
S4
12,13,14,15
S5
5,7,8,9
S6
16,17,18
(b)
START
S0
0,1,2,3,4,5,6,7,8,9,18
S1
3,4,5,6,7,8,9,18
S2
10,11
S3
19
S4
12,13,14,15
S5
5,7,8,9
S6
16,17,18
(c)

【図１６】



フロントページの続き

(72)発明者　スブラジット・バタチャリヤ
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内

(72)発明者　アナンド・ラグナサン
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内

(72)発明者　アーティ・グプタ
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内

Ｆターム(参考)　5B046 AA08 BA03 JA01 JA04